MEF UNIVERSITY

THE EFFECT OF EXCHANGE RATE VOLATILITY ON EXPORT AND IMPORT OF TURKEY ON SECTORAL BASIS

Capstone Project

Yağmur Ulutürk Tekten

İSTANBUL, 2018

MEF UNIVERSITY

THE EFFECT OF EXCHANGE RATE VOLATILITY ON EXPORT AND IMPORT OF TURKEY ON SECTORAL BASIS

Capstone Project

Yağmur Ulutürk Tekten

Advisor: Asst. Prof. Nazlı Toraganlı Karamollaoğlu

İSTANBUL, 2018

MEF UNIVERSITY

Name of the project: The Effect of Exchange Rate Volatility On Export and Import of Turkey On Sectoral Basis
Name/Last Name of the Student: Yağmur Ulutürk Tekten
Date of Thesis Defense: 03/09/2018

I hereby state that the graduation project prepared by Yağmur Ulutürk Tekten has been completed under my supervision. I accept this work as a "Graduation Project".

03/09/2018
Asst. Prof. Nazlı Toraganlı Karamollaoğlu

I hereby state that I have examined this graduation project by Yağmur Ulutürk Tekten which is accepted by his supervisor. This work is acceptable as a graduation project and the student is eligible to take the graduation project examination.

03/09/2018
Director of
Big Data Analytics Program
Prof. Özgür Özlük

We hereby state that we have held the graduation examination of Yağmur Ulutürk Tekten and agree that the student has satisfied all requirements.

THE EXAMINATION COMMITTEE

Committee Member	Signature
1. Asst. Prof. Nazlı Toraganlı Karamollaoğlu	……..
2. Prof. Özgür Özlük	……..

ACADEMIC HONESTY PLEDGE

I promise not to collaborate with anyone, not to seek or accept any outside help, and not to give any help to others.

I understand that all resources in print or on the web must be explicitly cited.

In keeping with MEF University's ideals, I pledge that this work is my own and that I have neither given nor received inappropriate assistance in preparing it.

YAĞMUR ULUTÜRK TEKTEN 03/09/2018

Name Date Signature

EXECUTIVE SUMMARY

THE EFFECT OF EXCHANGE RATE VOLATILITY ON EXPORT AND IMPORT OF TURKEY ON SECTORAL BASIS

Yağmur Ulutürk Tekten

Advisor: Asst. Prof. Nazlı Toraganlı Karamollaoğlu

SEPTEMBER, 2018, 76 pages

In this study, the effects of exchange rate volatility on export and import of Turkey is analysed by employing monthly trade data for the period from January 2004 to November 2015. The study is extended to cover both sectoral and country specific export and import volumes.

The major aim of this study is to show how fluctuations in foreign exchange rate change the volume of exports and imports among various sectors in Turkey. In this paper export and import volume equation is formulated using sectoral data in which explanatory variables are derived from the volatility of each country's nominal exchange rate against the TRY, bilateral real effective exchange rates for each country that Turkey has foreign trade relationship. The dependent or target variable is the percentage change in the trade size in USD amount both for export and import.

In this analysis, 6 different regression algorithms are utilized to explain the effect of exchange rate volatility on industrial activities for export and import in Turkey. The impact of features on the target feature is analyzed using linear, ridge, lasso, random forest, decision tree and gradient boosting regression algorithms.

According to results of these 6 algorithms, for Turkey, the volatility of exchange rate has significant impact on some sectors and on broad product group categories in both export and import up to 26%. The sectors that most exposed to exchange rate volatilities are seen in 'Giyim Eşyası' in export and 'Binek otomobilleri' in import.

For export, 'Rusya Federasyonu', and for import 'İtalya' is the most sensitive countries against exchange rate volatility in Turkey.

KeyWords: Import, Export, Exchange Rate Volatility, Real Exchange Rate, Regression, Ridge, Lasso

ÖZET

DÖVİZ KURUNDAKİ OYNAKLIĞIN SEKTÖREL BAZDA TÜRKİYE'NİN İHRACAT VE İTHALAT HACMİ ÜZERİNDEKİ ETKİSİ

Yağmur Ulutürk Tekten

Tez Danışmanı: Asst. Prof. Nazlı Toraganlı Karamollaoğlu

EYLÜL, 2018, 76 sayfa

Bu çalışmada döviz kurundaki oynaklığın Türkiye'nin farklı sektörlerinin ihracat ve ithalat hacimlerine olan etkisi incelenmiştir. Çalışmada kullanılan veri seti aylık bazda Ocak 2004 ve Kasım 2015 tarihlerini kapsamaktadır. Döviz kurundaki oynaklığın etkisi sektör ve ülke bazında hem ithalat hem ihracat alanlarında analiz edilmiştir.

Bu çalışmanın temel amacı son yıllarda artan döviz kurundaki dalgalanmaların Türkiye'de farklı sektörlerin ihracat ve ithalat hacimlerindeki değişimine olan etkisini anlamaya çalışmaktır.

Raporda, ihracat ve ithalat hacimlerindeki değişim sektör ve ürün grupları bazında formülize edilmiştir. Modellemede kullanılan bağımsız değişkenler Türk Lirasına karşılık çeşitli ülkelerin nominal döviz kurları, Türkiye ile ticari ilişkisi olan ülkelerin para birimleri ve Türk Lirası arasındaki reel efektif döviz kurları baz alınarak oluşturulmuştur. Modellemede kullanılan bağımlı değişkenler ise ihracat ve ithalat rakamlarındaki aylara göre yüzdesel değişimlerdir.

Bu çalışmada döviz kuru oynaklığının sektörlerin ihracat ve ithalat hacimlerindeki etkisi regresyon metodu ile analiz edildi. Bağımsız değişkenlerin modele katkısı linear, ridge, lasso, rassal orman, karar ağaçları ve gradyan artırma regresyon modelleriyle incelendi.

6 farklı regresyon algoritmasının sonuçlarına göre, Türkiye'de döviz kuru oynaklığının bazı sektör ve geniş ürün gruplarının ihracat ve ithalat hacimlerindeki değişimi üzerinde 26%' ya kadar ciddi etkileri olduğu saptanmıştır. Döviz kuru oynaklığından en çok etkilenen sektörün ihracatta 'Giyim eşyası', ithalatta ise 'Binek otomobilleri' olduğu gözlenmiştir. İhracatta 'Rusya Federasyonu' en fazla etkilenen ülke olurken, ithalatta ise 'İtalya' döviz kurundaki oynaklıktan en çok etkilenen ülke olarak öne çıkmıştır.

Anahtar Kelimeler: İthalat, İhracat, Döviz Kuru Oynaklığı, Reel Döviz Kuru, Regresyon, Ridge, Lasso

TABLE OF CONTENTS

1. INTRODUCTION

This study analyzes the impact of exchange rate volatility on the change of export and import in Turkey on sectoral basis. For Turkey, as a middle-income country, export growth and the composition of the goods in exports have a significant impact on the growth. In this respect, export sectors and the share of these sectors in Turkey's export play a key role in terms of the growth performance of Turkey.

The objective of this paper is to understand how the exchange rate volatility affects the change of exports and imports across Turkey's different sectors. The use of trade volume data at country level may lead to the aggregation bias problem since the effect of exchange rate volatility on export and import may vary across sectors in a country (Bahmani-Oskooee and Wang 2008). For this reason, employing the sector-level export and import volume data in order to deal with the aggregation bias problem is preferred to analyse the impact of exchange rate volatility on sector-level trade.

1.1. About the Data

The export and import volume based on cross-sectoral and cross-country specific data is obtained from Turkish Statistical Institute (TUIK) online trade database. The nominal exchange rates are retrieved from the Electronic Data Distribution Center of The Central Bank of the Republic of Turkey (TCMB EVDS). The country specific price indexes for computation of real exchange rates are obtained from the OECD statistics library.

The dataset covers trade statistics over the period between January 2004 and November 2015. The export dataset contains more than 1.9 million rows with 10 variables, while import dataset has more than 1.2 million rows with 10 variables.

The sectoral categorization in the TUIK database named as ISIC3_CODE complies with classification of product group by technology intensity prepared by OECD standards. Therefore, ISIC3_CODE refers to this classification standards in the dataset.

BEC column in the dataset refers to the BEC classification (Classification by Broad Economic Categories) that provides a means for international trade statistics to be analyzed by broad economic categories such as food, industrial supplies, capital equipment, consumer durables and consumer non-durables. Thus, BEC is a goods classification of foreign trade

statistics. The classification correlates the goods to broad macroeconomic categories such as capital goods, intermediates goods, and consumer goods.

1.2. Project Definition

The study analyzes the impact of exchange rate volatility on Turkey's export and import for various sectors. The major aim of the study is to show variability in the changes in export and import volume on sectoral basis against the volatility in the foreign exchange rates.

1.3. Methodology

The model is formulated as a regression problem. The contributions of explanatory variables are analysed using regression analysis on time series trade volume data and foreign exchange rates.

There are 16 explanatory variables in the modelling. These variables are:

	Features	Feature_Type	Explanation
1	YEAR	Feature	Year in numerical format
2	MONTH	Feature	Month in numerical format
3	ISIC3	Feature	ISIC3 code. Categorical values transformed into integers by label encoding method.
4	BEC	Feature	BEC code. Categorical values transformed into integers by label encoding method.
5	COUNTRY	Feature	COUNTRY code. Categorical values transformed into integers by label encoding method.
6	REXR	Feature	Calculated field. Bilateral real exchange rate for two countries
7	CLOSE-OPEN_STD	Feature	6 month window standard deviation of percentage change in 'Close-Open Prices'
8	HIGH-LOW_STD	Feature	6 month window standard deviation of percentage change in 'High-Low Prices'
9	CLOSE_STD	Feature	6 month window standard deviation of percentage change in 'Close Prices'
10	REXR_STD	Feature	6 month window standard deviation of percentage change in 'Real Exchange Rates'
11	REXR_PCT_1	Feature	Percentage change in REXR for 1 month window
12	REXR_PCT_3	Feature	Percentage change in REXR for 3 month window
13	REXR_PCT_6	Feature	Percentage change in REXR for 6 month window
14	CLOSE_PCT_1	Feature	Percentage change in CLOSE PRICE for 1 month window
15	CLOSE_PCT_3	Feature	Percentage change in CLOSE PRICE for 3 month window
16	CLOSE_PCT_6	Feature	Percentage change in CLOSE PRICE for 6 month window
17	DOLLAR_AMOUNT_PCT_1	Target Feature	Percentage change in Dollar Amount for 1 month window
18	DOLLAR_AMOUNT_PCT_3	Target Feature	Percentage change in Dollar Amount for 3 month window
19	DOLLAR_AMOUNT_PCT_6	Target Feature	Percentage change in Dollar Amount for 6 month window

2. LITERATURE REVIEW

There is a wide range of academic studies that analyzes the change in export and import against the exchange rate volatility. Majority of the studies focuses on export income elasticity for different countries, while remaining ones try to model export and import demand function in order to define explanatory variables for export and import demand functions.
While some studies suggest that the exchange rate volatility has a negative impact on the export volume, some others argue the opposite recommending that the volatility on the exchange rate has a positive impact on the export size.

Berument et al. (2013) shows that income elasticity is greater than 1 for aggregate exports, whereas they suggest that the real exchange rate elasticity is less than 1. In many sectors, they found that income elasticity is greater than 1 with significant variations across countries. Nazlioglu (2012) states that the impact of the exchange rate volatility on Turkish exports differs across industries and the depreciation of the Turkish lira has positive impacts on Turkish industry-level exports. Demez and Ustaoglu (2012) show that export is not sensitive to the volatility in currency rates by applying Zivot-Andrews' unit root test with one structural break and Lee-Strazicich' s unit root test with two structural breaks methodologies.

Demirhan (2015) shows that exchange-rate stability has a significant positive effect on real export volume, both in the short and the long run. In this study, The Johansen multivariate cointegration method and the parsimonious error-correction model are used to determine long-run and short-run relationships between real export volume and its determinants. In addition, GARCH model is taken as a proxy for exchange-rate stability and generalized impulse-response functions and VAR analyses are employed to analyze the effects of variables on real export volume.

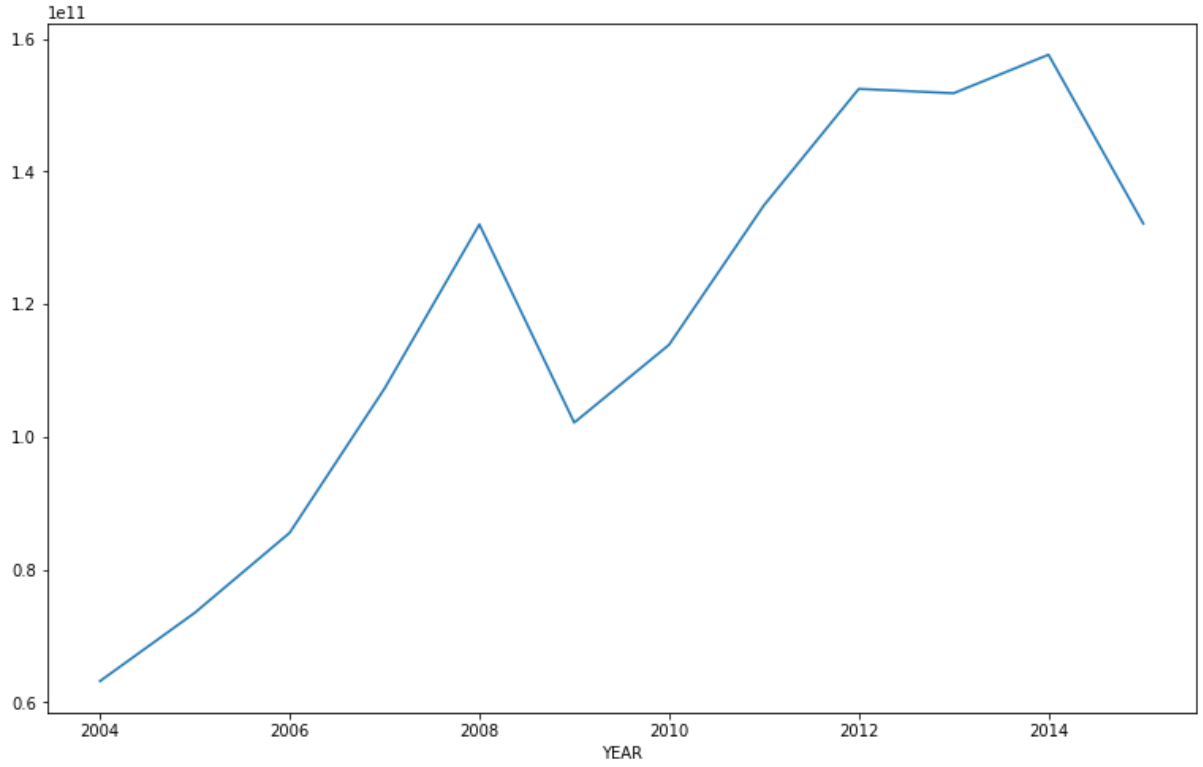
Tari and Yildirim (2009) employ error correction and Johansen cointegration models and suggest that volatility of exchange rate influences export volume negatively in the long run. However, the uncertainty of exchange rate does not have an impact on export volume in the short run.

In contrast to the previous studies, Altintas et al. (2011) shows that relative prices have a negative effect and foreign income has an insignificant effect. However, nominal exchange

rate volatility has a positive and significant effect on Turkish exports by applying multivariate cointegration and error correction model techniques. This study also suggests that disaggregated trade data which means industry-level trade data should be analyzed in order to make clear policy recommendations.

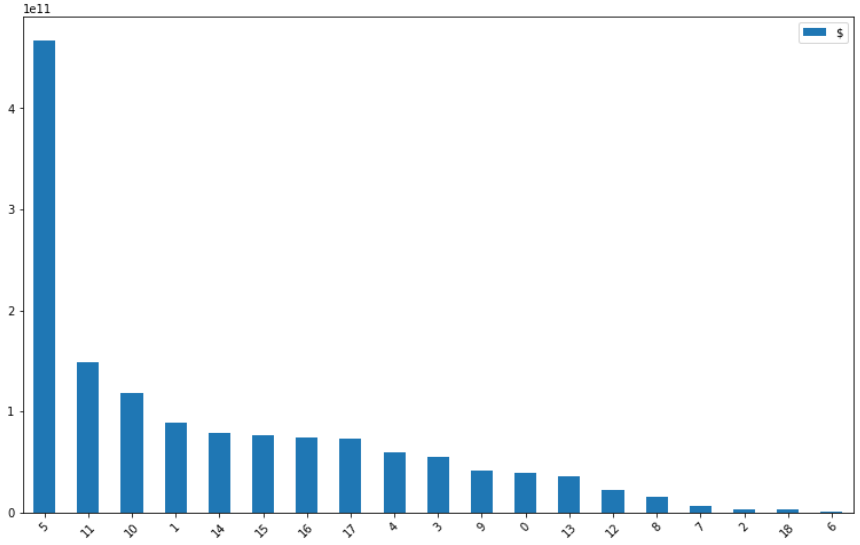
3. DESCRIPTIVE ANALYSIS ON EXPORT DATASET

3.1. Turkey's Export over Years (2004-2015)



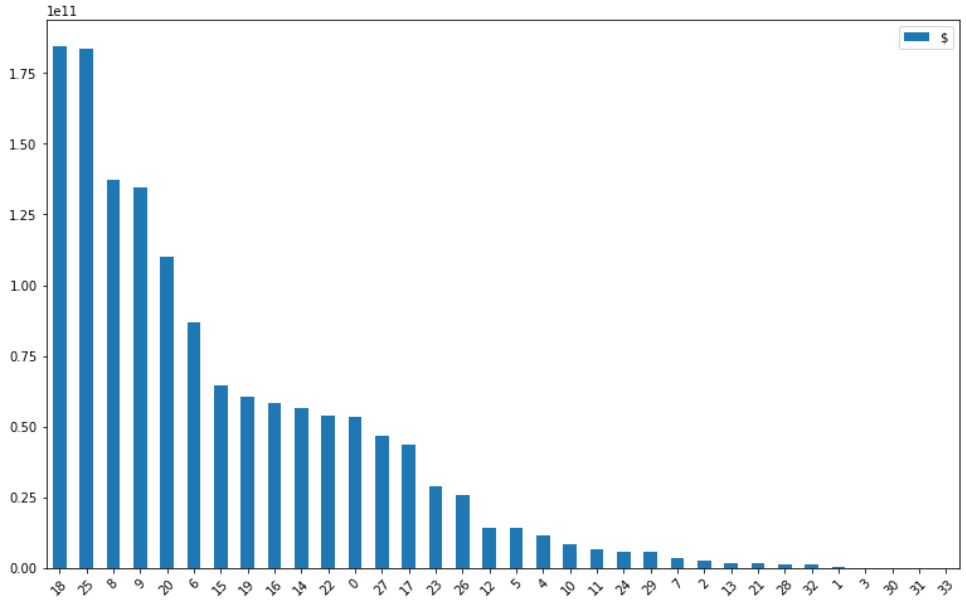
3.2. Turkey's Export by Sectoral Breakdown (2004-2015 Total)

BEC categorization



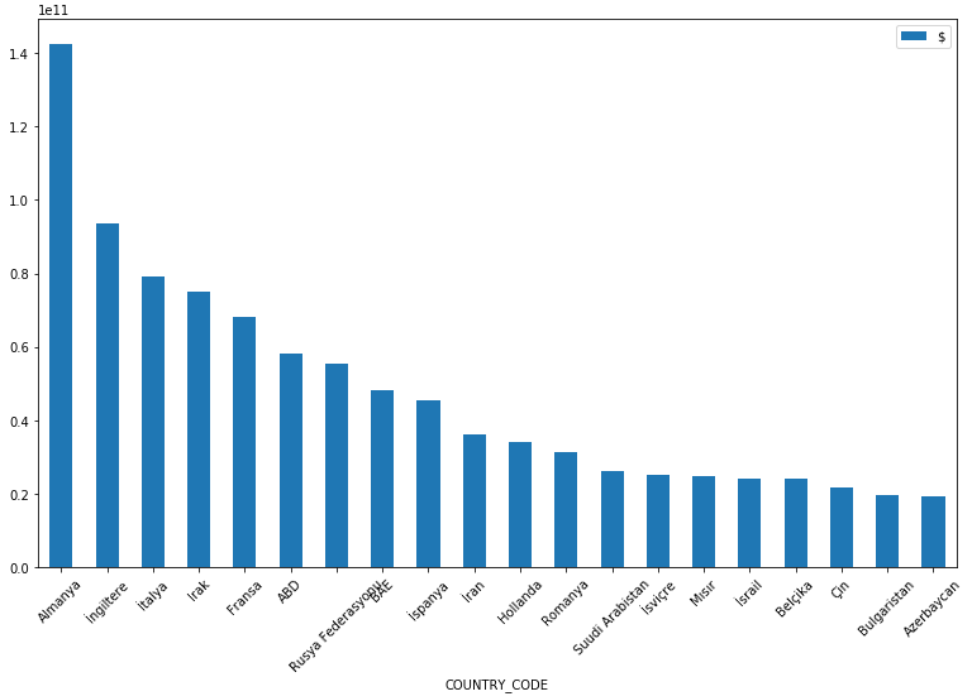
3.3. Turkey's Export by Sub-Sectoral Breakdown (2004-2015 Total)

ISIC3 categorization



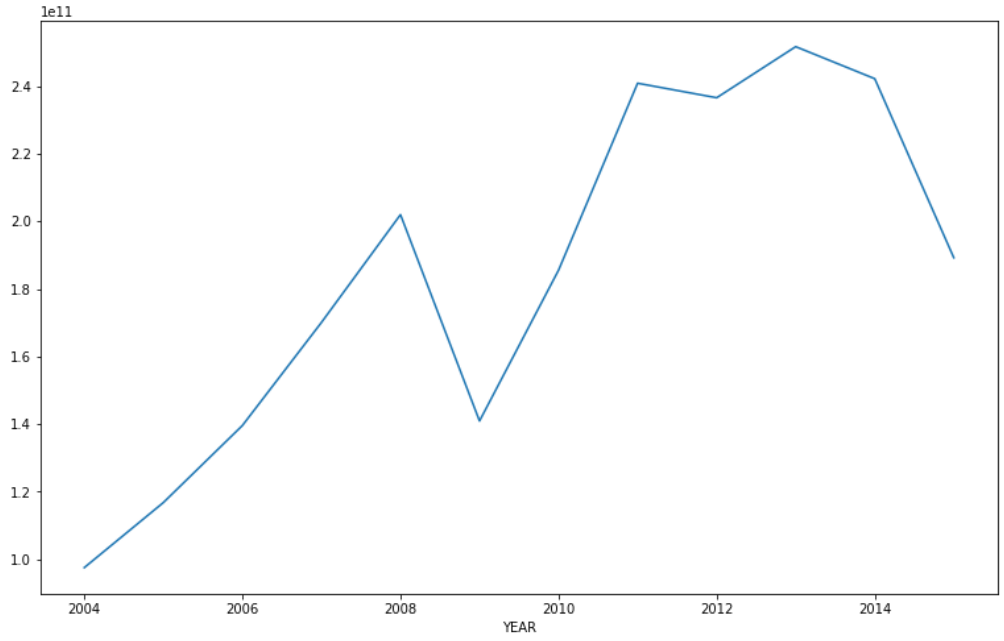
3.4. Turkey's Export by Trade Partners (2004-2015 Total)

Country level categorization



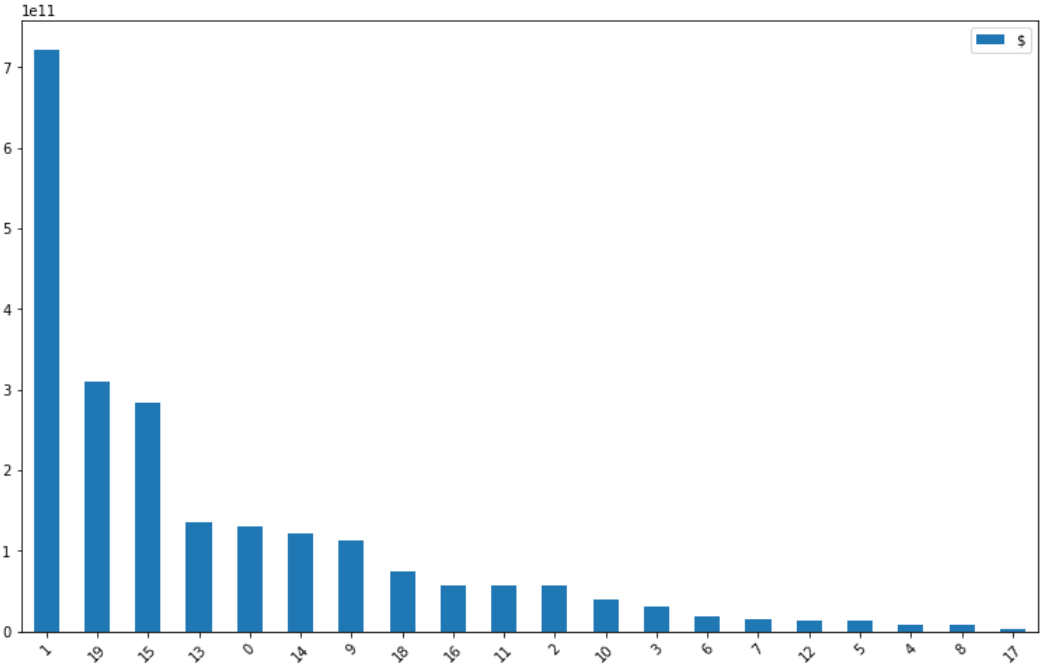
4. DESCRIPTIVE ANALYSIS ON IMPORT DATASET

4.1. Turkey's Import over Years (2004-2015)



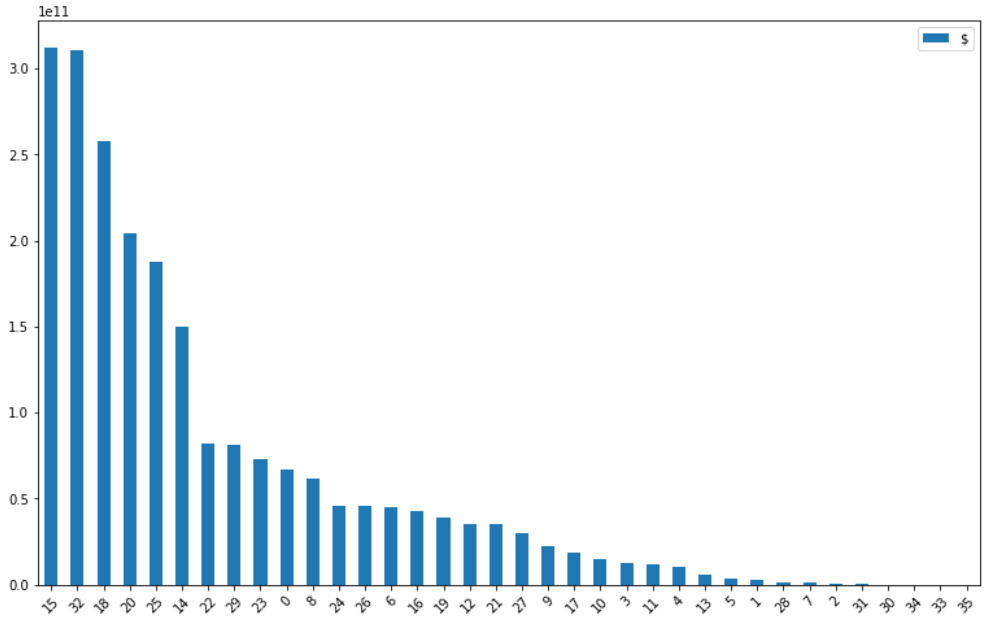
4.2. Turkey's Import by Sectoral Breakdown (2004-2015 Total)

BEC categorization



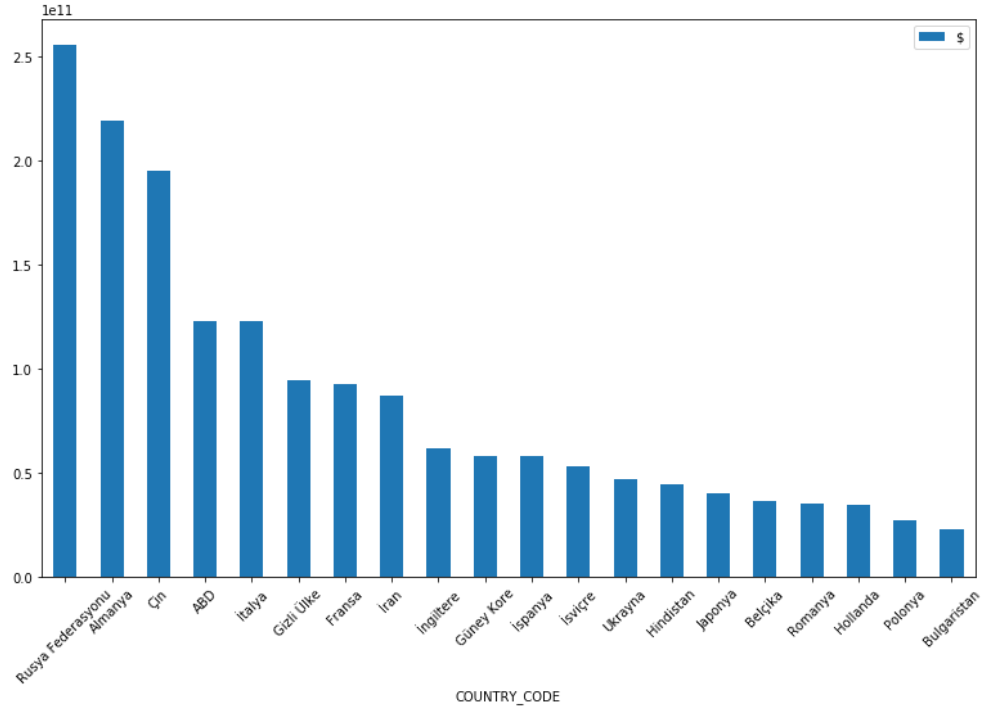
4.3. Turkey's Import by Sub-Sectoral Breakdown (2004-2015 Total)

ISIC3 categorization



4.4. Turkey's Import by Trade Partners (2004-2015 Total)

Country level categorization



5. DATA PREPROCESSING & FEATURE ENGINEERING

5.1. Data Cleaning

Firstly, the original dataset is divided into two parts which are import and export datasets. Then, all of the data preprocessing tasks are performed on both of them. Data cleaning process begins with the detecting duplicate codes for same countries. The countries with the same country codes are detected and replaced with suitable values. However, duplicate country codes problem does not exist in the export dataset. Afterwards the rows which all values have NAN are dropped both for import and export datasets. Import data frames for different time ranges are merged.

We select observations from 2004 and afterwards, since export dataset lacked "MONTH" column before 2004. Another reason for such a decision is that before European Union, there were lots of different currencies and exchange rate combinations which would increase complexity in the model.

In the dataset, there are countries using more than one currency pair in export and import in their trade relationships. In order to reduce currency complexity, for each country only one currency pair is selected based on conventional nominal exchange rates. We also need to sum trade volumes for such cases, otherwise we would lose information.

Then each country code and each currency code are matched and merged on the import and export datasets. There are 34 countries in the datasets but not 34 unique currencies since some countries use common currencies such as those in European Union use EUR in international trade.

There are some missing values in Currency Names. These cases are actually had "US Dollar Code" (400) in Currency Code cells. Thus, Currency Names of such cases are filled with "US Dollar".

CURRENCY_PAIR	CURRENCY_CODE
audtry	1
brltry	2
cadtry	3
chftry	4
cnytry	5
czktry	6
dkktry	7
eurtry	8
gbptry	9
hkdtry	10
ilstry	11
inrtry	12
jpytry	13
mxntry	14
noktry	15
nzdtry	16
plntry	17
rontry	18
rubtry	19
sartry	20
sektry	21
usdtry	22
zartry	23

Mapping data frames for ISIC3, BEC, Currency and Country codes and names are created. Rows which do not have information for currency or any missing values are dropped. Lastly, data types of pandas data frame columns are corrected for any future error.

	COUNTRY_NAME	COUNTRY_CODE	CURRENCY_PAIR	CURRENCY_CODE		COUNTRY_NAME	COUNTRY_CODE	CURRENCY_PAIR	CURRENCY_CODE
1	ABD	400	usdtry	22	18	İspanya	11	eurtry	8
2	Almanya	4	eurtry	8	19	İsrail	624	ilstry	11
3	Avustralya	800	audtry	1	20	İsveç	30	sektry	21
4	Avusturya	38	eurtry	8	21	İsviçre	36	chftry	4
5	Belçika	17	eurtry	8	22	İtalya	5	eurtry	8
6	Brezilya	508	brltry	2	23	Japonya	732	jpytry	13
7	Çek Cumhuriyeti	61	czktry	6	24	Kanada	404	cadtry	3
8	Çin	720	cnytry	5	25	Malta	46	eurtry	8
9	Danimarka	8	dkktry	7	26	Meksika	412	mxntry	14
10	Fransa	1	eurtry	8	27	Norveç	28	noktry	15
11	Güney Afrika	390	zartry	23	28	Polonya	60	plntry	17
12	Hindistan	664	inrtry	12	29	Portekiz	10	eurtry	8
13	Hırvatistan	92	eurtry	8	30	Romanya	66	rontry	18
14	Hollanda	3	eurtry	8	31	Rusya Federasyonu	75	rubtry	19
15	Hong Kong	740	hkdtry	10	32	Suudi Arabistan	632	sartry	20
16	İngiltere	6	gbptry	9	33	Yeni Zelanda	804	nzdtry	16
17	İrlanda	7	eurtry	8	34	Yunanistan	9	eurtry	8

After cleaning duplicate and missing values for currencies and countries, monthly consumer price indices (CPI) are added to the datasets. Having imported Turkey's and other countries' monthly CPI values, only those countries which we are able to import monthly CPI values are kept for the rest of the study. The reason is CPI values are necessary to calculate bilateral real exchange rates between Turkey and other countries. The countries which do not have monthly CPI values are dropped from both datasets. A mapping data frame is generated for matching countries' names in English, Turkish and related 'COUNTRY CODE'.

5.2. Feature Extraction

After merging CPI values with the export and import datasets, nominal exchange rates for those different countries against TRY are added to the datasets. There are 23 currency pairs against TRY. The monthly series of those 23 nominal exchange rates between 2004-2015 years are imported from the website called 'Investing.com'. There are Close, Open, High and Low prices for nominal exchange rates on monthly basis. When nominal exchange rates are merged with the export and import datasets, monthly real exchange rates between Turkey and other countries are calculated and a new feature called 'REXR' is added to the datasets. The real exchange rates between two countries are calculated as below:

*Real Exchange Rate = Nominal Exchange Rate * CPI (domestic) / CPI (foreign)*

The other feature named 'CLOSE-OPEN' which refers to the price range (close price minus open price) within a month is created and added to the datasets. The third one called 'HIGH-LOW' stands for the difference between the HIGH and LOW price that is observed within a month.

After including those currency related features, the datasets which have already been in the form of Python data frame are turned into pandas time-series format using 'to_period' method. This transformation was necessary in order to calculate percentage changes and standard deviations of changes in the exchange rates, since the data frames should be in time-series format. Otherwise, new features would not be generated from existing ones like High, Low, Close, Open prices.

Pandas ‘to_period’ method was useful in this case since one can easily convert pandas data frame from DatetimeIndex to PeriodIndex with desired frequency. In our case all datasets are in monthly frequency in DateTimeIndex format and they are transformed into monthly periods using ’to_period’ method.

An example script about how ‘to_period’ method is utilized in this case:

```
# Convert dfM & dfX to Pandas time-series format
dfM['MONTH_YEAR'] = dfM['MONTH'].map(str) + '/' + dfM['YEAR'].map(str)
dfM['MONTH_YEAR'] = pd.to_datetime(dfM['MONTH_YEAR'], errors='raise', yearfirst=True, format='%m/%Y').dt.to_period('M')
dfM['SUPER_INDEX'] = dfM['COUNTRY_CODE'].map(str) + '/' + dfM['ISIC3_CODE'].map(str) + '/' + dfM['BEC_CODE'].map(str)
dfM.set_index(['MONTH_YEAR','SUPER_INDEX'], drop=False, append=False, inplace=True, verify_integrity=False)
dfM.sort_index(axis=0, level=[0,1], ascending=[True,True], inplace=True, sort_remaining=False)

dfX['MONTH_YEAR'] = dfX['MONTH'].map(str) + '/' + dfX['YEAR'].map(str)
dfX['MONTH_YEAR'] = pd.to_datetime(dfX['MONTH_YEAR'], errors='raise', yearfirst=True, format='%m/%Y').dt.to_period('M')
```

It was easy to calculate percentage changes and also standard deviations in the nominal exchange rates’ percentage changes, when we have a data frame in pandas time-series format. For the features REXR, CLOSE-OPEN, HIGH-LOW and CLOSE 6-month period standard deviations of percentage changes are calculated, while for the REXR and CLOSE percentage changes for 1, 3 and 6 month periods are calculated separately using pandas' *stack/unstack, pct_change* and *rolling* methods. These new features are included to the export and import data frames.

After including calculated features such as REXR_STD, REXR_PCT_1 or CLOSE_STD, some rows are filled with NAN due to lack of values in the previous periods. Those rows and features on 13 columns with NAN values are dropped before running regression models.

At this step, feature list is extended with 17 new features as seen in the table below:

1	CLOSE
2	OPEN
3	HIGH
4	LOW
5	CLOSE-OPEN
6	HIGH-LOW
7	REXR
8	CLOSE_STD
9	CLOSE-OPEN_STD
10	HIGH-LOW_STD
11	REXR_STD
12	CLOSE_PCT_1
13	CLOSE_PCT_3
14	CLOSE_PCT_6
15	REXR_PCT_1
16	REXR_PCT_3
17	REXR_PCT_6

5.3. Label Encoding

Before executing regression model, categorical variables ISIC3_CODE, BEC_CODE, COUNTRY_CODE should be turned into numerical format. Those categorical variables are transformed into numerical format using label encoding method. Label encoding simply converts each value in the column to a number that stands for the index of the column. For label encoding, 'cat_codes' function is applied to those three features. At the final stage, the types of features are turned into numerical format (float or integer) that are ready to be used in the regression analysis.

```
YEAR               int64
MONTH              int64
REXR             float64
REXR_PCT_1       float64
REXR_PCT_3       float64
REXR_PCT_6       float64
CLOSE_STD        float64
CLOSE-OPEN_STD   float64
HIGH-LOW_STD     float64
REXR_STD         float64
CLOSE_PCT_1      float64
CLOSE_PCT_3      float64
CLOSE_PCT_6      float64
ISIC3               int8
BEC                 int8
COUNTRY             int8
```

5.4. Outliers

When we look at the distribution of these new features, we can easily infer that there are outlier values that need to be manipulated. Excluding those outlier values from the datasets that will be used in the modelling was necessary. For this reason threshold values are defined for each feature according to their histograms and those thresholds are used as parameters in the function below:

```
def remove_outliers(df_man, col, threshold):
    df = df_man.copy()
    del_df = df[df[col] > threshold]
    print('There are %d outliers in the %s out of %d entries' % (len(del_df), str(col), len(df)))
    df = df[df[col] <= threshold]
    return df

dfM_model = remove_outliers(dfM_model, 'DOLLAR_AMOUNT_PCT_1', 1)
dfM_model = remove_outliers(dfM_model, 'DOLLAR_AMOUNT_PCT_3', 1)
dfM_model = remove_outliers(dfM_model, 'DOLLAR_AMOUNT_PCT_6', 1)
dfM_model = remove_outliers(dfM_model, 'REXR', 200000)
dfM_model = remove_outliers(dfM_model, 'CLOSE-OPEN_STD', 90)
dfM_model = remove_outliers(dfM_model, 'HIGH-LOW_STD', 90)
dfM_model = remove_outliers(dfM_model, 'CLOSE-OPEN_STD', 90)

dfX_model = remove_outliers(dfX_model, 'DOLLAR_AMOUNT_PCT_1', 1)
dfX_model = remove_outliers(dfX_model, 'DOLLAR_AMOUNT_PCT_3', 1)
dfX_model = remove_outliers(dfX_model, 'DOLLAR_AMOUNT_PCT_6', 1)
dfX_model = remove_outliers(dfX_model, 'REXR', 200000)
dfX_model = remove_outliers(dfX_model, 'CLOSE-OPEN_STD', 90)
dfX_model = remove_outliers(dfX_model, 'HIGH-LOW_STD', 20)
dfX_model = remove_outliers(dfX_model, 'CLOSE-OPEN_STD', 90)
```

5.5. Feature Scaling

The feature named ‘YEAR’ refers to year in number and it is scaled just by deducting 2003 from each year value since there are 12 years in the datasets that begin with 2004 and end with 2015. Through this simple scaling method, each year is represented as 1 to 12 instead of 2004 to 2015.

Year	Year after Scaling
2004	1
2005	2
2006	3
2007	4
2008	5
2009	6
2010	7
2011	8
2012	9
2013	10
2014	11
2015	12

The features ‘HIGH-LOW_STD’ and ‘CLOSE-OPEN_STD’ are also needed to be scaled. These two features are scaled with MinMaxScaler. After attempting several feature scaling methods (MinMaxScaler, StandardScaler, MaxAbsScaler, RobustScaler, Normalizer) to those two features, the best scaler method is seen as MinMaxScaler.

Before starting model building, we save pickle files of import and export features and targets, in order to use them in Azure Machine Learning Studio for BDA 564 Final Project.

6. MODEL BUILDING

In this section, export and import equation models, regression analysis output and model parameters will be shown. Cross country level and cross-sectoral level coefficients and relationships among the model variables will be analyzed in this section.

6.1. Dependent Variable

For the regression model, three different target features or independent variables are created. These are:

Target Feature	Definition
DOLLAR_AMOUNT_PCT_1	Percentage change in Dollar Amount for 1 month window
DOLLAR_AMOUNT_PCT_3	Percentage change in Dollar Amount for 3 month window
DOLLAR_AMOUNT_PCT_6	Percentage change in Dollar Amount for 6 month window

For the export and import datasets, instead of absolute values of USD trade amount, percentage changes in 1, 3 and 6 month-periods are set as target features. Each regression model tried to explain one of these 3 targets each time. In other words, we do not try to predict DOLLAR_AMOUNT_PCT_1, DOLLAR_AMOUNT_PCT_3, DOLLAR_AMOUNT_PCT_6 at the same time.

6.2. Independent Variables

In the final datasets both for import and export, there are 16 explanatory features that are used in the regression models. All of these features are used in the model. 16 features in the model are listed in the table below:

	Features	Feature_Type	Explanation
1	YEAR	Feature	Year in numerical format
2	MONTH	Feature	Month in numerical format
3	ISIC3	Feature	ISIC3 code. Categorical values transformed into integers by label encoding method.
4	BEC	Feature	BEC code. Categorical values transformed into integers by label encoding method.
5	COUNTRY	Feature	COUNTRY code. Categorical values transformed into integers by label encoding method.
6	REXR	Feature	Calculated field. Bilateral real exchange rate for two countries
7	CLOSE-OPEN_STD	Feature	6 month window standard deviation of percentage change in 'Close-Open Prices'
8	HIGH-LOW_STD	Feature	6 month window standard deviation of percentage change in 'High-Low Prices'
9	CLOSE_STD	Feature	6 month window standard deviation of percentage change in 'Close Prices'
10	REXR_STD	Feature	6 month window standard deviation of percentage change in 'Real Exchange Rates'
11	REXR_PCT_1	Feature	Percentage change in REXR for 1 month window
12	REXR_PCT_3	Feature	Percentage change in REXR for 3 month window
13	REXR_PCT_6	Feature	Percentage change in REXR for 6 month window
14	CLOSE_PCT_1	Feature	Percentage change in CLOSE PRICE for 1 month window
15	CLOSE_PCT_3	Feature	Percentage change in CLOSE PRICE for 3 month window
16	CLOSE_PCT_6	Feature	Percentage change in CLOSE PRICE for 6 month window

6.3. Modelling and Parameters

In this study, 6 different regression algorithms are used for both export and import datasets. These algorithms are:

I. Linear Regression
II. Ridge Regression
III. Lasso Regression
IV. Random Forest Regression
V. Decision Tree Regression
VI. Gradient Boosting Regression

We define a regression function which takes "features", "target", "explanation", "trade_type" and "algo" as input parameters, and generates results for given parameters. The reason behind defining such a function is that we will have multiple regression analysis for each trade type (import or export), each country or sector, each target or each regression algorithm (ex: Linear Regression or an ensemble method). We call regression function each time we have a different combination of input parameters. Combining all these different regression models, we generate more than 1.500 results. The results are saved into "results_df" data frame and stored into a CSV file, so we can easily access the results for further analysis. The import and export datasets have more than 160.000 observations with 16 features. model.py script takes 5 minutes in order to execute all the code.

Since there are more than 30 countries and a great deal amount of sectors (ISIC3) and broad product group classifications (BEC), there should be a lot of distinct regression models that take into account those sector and product group breakdowns. This was a tricky part for the analysis since when we split the dataset into breakdowns, some sector or product group breakdowns have even less than 10 observations. Having insufficient number of observations for regression models would have produced biased and erroneous results. For this reason, we add minimum sample size parameter into our regression function. Minimum sample size is set as 100 (*min_samples = 100*) in the regression function so that when a dataset for different categorical feature breakdown has less than 100 samples, the regression output will return 0

for all performance metrics. Otherwise, the given regression output will be generated and printed.

Apart from minimum sample size, this new function takes into account other tuning parameters as well. Those parameters are explained as below:

- **The test-train split:** Test size fraction is set to be 0.30. for both import and export.
- **Random state parameter:** It is given 7 for all of the models.
- **Ridge Regression:** Alpha parameter is set to 1.0.
- **Random Forest Regression:** Alpha is given 0.1.

After executing all regressions, we end up with a great deal amount of regression outputs that makes difficult to understand and interpret the results. The regression function outputs are stored so that we can see all the results for each combination. The results, output metrics and regression coefficients are shown in a data frame in human readable and comprehensible format. The final data frame for results contains the 28 columns in the table below:

Trade_Type	Coef_REXR
Breakdown_Type'	Coef_REXR_PCT_1
Breakdown_Code	Coef_REXR_PCT_3
Breakdown_Name	Coef_REXR_PCT_6
Explanation	Coef_CLOSE_STD
Target	Coef_CLOSE-OPEN_STD
Algorithm	Coef_HIGH-LOW_STD
MSE_Test	Coef_REXR_STD
R2_Test	Coef_CLOSE_PCT_1
MSE_Train	Coef_CLOSE_PCT_3
R2_Train	Coef_CLOSE_PCT_6
Sample_Size	Coef_ISIC3
Coef_YEAR	Coef_BEC
Coef_MONTH	Coef_COUNTRY

When all regression functions are executed, one of the ISIC3 code ‘33’ and BEC code ‘17’ in import dataset outperformed other models by R2 scores with higher than 40%. These R2 scores were extremely higher than the other regression results for other country, BEC and

ISIC3 breakdowns. The sector with ISIC3_33 and the BEC_17 refer to 'Gizli Veri', i.e, confidential information. These two codes were excluded from the final regression results. In the result data frame, breakdown names are added instead of codes for readability.

7. RESULTS & CONCLUDING REMARKS

In order to empirically analyze the impact of exchange rate volatility on industry-level and country-level trade, the regressions are executed on the import and export data for Turkey. Since the main goal of this study is to understand the effect of exchange rate volatility on trade volume, explanatory analysis is conducted for import and export datasets. Various regression algorithms are utilized to explain whether or not the exchange rate volatility has significant impact on the Turkey's export and import volume for different industrial activities.

6 different regression algorithms are applied and coefficients are stored for more than 1500 different regression functions in such a way that the contributions of each feature in each regression function can easily be interpreted.

When we look at the results table, we can easily read the regression outputs and make comments on the breakdowns. However, as a conclusion of the best five models for export and import, respectively, will be summarized below. The 6 regression models' performance will also be analyzed at the end of this section.

Regression Results Table:

Trade_Type	Breakdown_Type	Breakdown_Code	Breakdown_Name	Explanation	Target	Algorithm	MSE_Test	R2_Test	MSE_Train	R2_Train	Sample_Size
export	ISIC3	10	Giyim eşyası	ISIC3_10	DOLLAR_AMOUNT_PCT_6	randomforest	0.125	0.264	0.037	0.786	7526
export	ISIC3	10	Giyim eşyası	ISIC3_10	DOLLAR_AMOUNT_PCT_6	gradientboosting	0.127	0.253	0.114	0.330	7526
export	ISIC3	10	Giyim eşyası	ISIC3_10	DOLLAR_AMOUNT_PCT_3	randomforest	0.126	0.247	0.035	0.792	7526
import	BEC	6	Binek otomobilleri	BEC_6	DOLLAR_AMOUNT_PCT_1	gradientboosting	0.164	0.217	0.096	0.530	1454
import	BEC	6	Binek otomobilleri	BEC_6	DOLLAR_AMOUNT_PCT_3	gradientboosting	0.171	0.217	0.090	0.573	1454
export	BEC	12	Esası yiyecek ve içecek olan işlenmemiş tüketim malları	BEC_12	DOLLAR_AMOUNT_PCT_6	gradientboosting	0.184	0.213	0.159	0.304	5728
import	ISIC3	25	Motorlu kara taşıtı ve römorklar	ISIC3_25	DOLLAR_AMOUNT_PCT_3	gradientboosting	0.155	0.206	0.131	0.317	5680
export	COUNTRY	19	Rusya Federasyonu	COUNTRY_19	DOLLAR_AMOUNT_PCT_6	gradientboosting	0.191	0.202	0.177	0.276	6448
import	COUNTRY	3	İtalya	COUNTRY_3	DOLLAR_AMOUNT_PCT_6	gradientboosting	0.158	0.199	0.142	0.279	8446
import	ISIC3	20	Başka yerde sınıflandırılmamış makine ve teçhizat	ISIC3_20	DOLLAR_AMOUNT_PCT_3	gradientboosting	0.160	0.191	0.157	0.230	11902
export	ISIC3	10	Giyim eşyası	ISIC3_10	DOLLAR_AMOUNT_PCT_3	gradientboosting	0.135	0.191	0.120	0.293	7526
export	BEC	12	Esası yiyecek ve içecek olan işlenmemiş tüketim malları	BEC_12	DOLLAR_AMOUNT_PCT_6	randomforest	0.190	0.190	0.045	0.803	5728
import	BEC	18	Sanayi ile ilgili taşımacılık araç ve gereçleri	BEC_18	DOLLAR_AMOUNT_PCT_3	gradientboosting	0.217	0.190	0.136	0.488	1665
export	ISIC3	0	Tarım ve hayvancılık	ISIC3_0	DOLLAR_AMOUNT_PCT_6	gradientboosting	0.219	0.182	0.190	0.265	5206
import	ISIC3	20	Başka yerde sınıflandırılmamış makine ve teçhizat	ISIC3_20	DOLLAR_AMOUNT_PCT_6	gradientboosting	0.182	0.182	0.167	0.231	11902
export	ISIC3	9	Tekstil ürünleri	ISIC3_9	DOLLAR_AMOUNT_PCT_6	randomforest	0.147	0.181	0.050	0.712	14065
export	ISIC3	9	Tekstil ürünleri	ISIC3_9	DOLLAR_AMOUNT_PCT_6	gradientboosting	0.147	0.180	0.135	0.224	14065
import	BEC	18	Sanayi ile ilgili taşımacılık araç ve gereçleri	BEC_18	DOLLAR_AMOUNT_PCT_1	gradientboosting	0.236	0.176	0.147	0.458	1665
export	COUNTRY	19	Rusya Federasyonu	COUNTRY_19	DOLLAR_AMOUNT_PCT_6	randomforest	0.197	0.175	0.072	0.706	6448
import	BEC	6	Binek otomobilleri	BEC_6	DOLLAR_AMOUNT_PCT_3	randomforest	0.180	0.174	0.034	0.840	1454
import	ISIC3	11	Ağaç ve mantar ürünleri (mobilya hariç); hasır vb. örülerc	ISIC3_11	DOLLAR_AMOUNT_PCT_6	gradientboosting	0.197	0.173	0.158	0.361	3152
export	COUNTRY	7	Yunanistan	COUNTRY_7	DOLLAR_AMOUNT_PCT_6	gradientboosting	0.182	0.173	0.167	0.226	6608
export	ISIC3	26	Motorlu kara taşıtı ve römorklar	ISIC3_26	DOLLAR_AMOUNT_PCT_6	gradientboosting	0.169	0.172	0.148	0.256	7475

EXPORT

- The first best three R2 test score belongs to the models based on **ISIC3 = 'Giyim Eşyası'** breakdown in export.
- The first best two model has the target feature 'Dollar_Amount_PCT_6'.
- When we compare the best two models, we see that **Random Forest** regressor in 'Giyim Eşyası' with R2 score on test data 0.264 performs better than **Gradient Boosting** for export volume.
- The third best model has the target feature 'Dollar_Amount_PCT_3'. In this model, changes in the export volume of the 'Giyim Eşyası' against the exchange rate volatility is best explained with **Random Forest** regressor with 0.247 R2 score on test data.
- The successive best model in export is the one in which the target feature is again 'Dollar_Amount_PCT_6', but the breakdown type is BEC.
- When regression is run on **BEC = 'Esası yiyecek ve içecek olan işlenmemiş tüketim malları'** for export dataset, we can see that **Gradient Boosting** regressor best explains the 6 month percentage change in the dollar amount of export volume in 'Esası yiyecek ve içecek olan işlenmemiş tüketim malları' product category against the volatility in exchange rates.
- The other best performance model in export is seen when the target feature is again 'Dollar_Amount_PCT_6' and breakdown type is COUNTRY.
- For 'Rusya Federasyonu', **Gradient Boosting** regressor best explains the 6 month percentage change in the dollar amount of export volume against the volatility in exchange rates with R2 test score 0.202

IMPORT:

- The best performance model for import is seen when the target feature is 'Dollar_Amount_PCT_1' and the breakdown type is BEC.
- Gradient Boosting regressor best explains the changes in the 1 month percentage change in the import volume in BEC = 'Binek otomobilleri' with R2 score on test data 0.217

- The second best performance model in import is again attained with Gradient Boosting when BEC is again ‘Binek otomobilleri’, but in this case the target feature is ‘Dollar_Amount_PCT_3’
- The third best model in import is seen with Gradient Boosting regressor when breakdown type is ISIC3 = ‘Motorlu kara taşıtı ve römorklar’. Gradient Boosting regressor performs better when the target feature is again ‘Dollar_Amount_PCT_3’ and ISIC3 breakdown is set to be ‘Motorlu kara taşıtı ve römorklar’. At this model, R2 score on test data is 0.206.
- The fourth best model in import is seen when breakdown is set on COUNTRY.
- The Gradient Boosting regressor seems the best when we try to analyze the impact of exchange rate volatility on 6 month percentage change in the dollar amount of imports from Italy with R2 score on test set is 0.199.
- The fifth best model in import is obtained when breakdown is set on ISIC3. Again, Gradient Boosting regressor performs better for import.
- In the fifth one, Gradient Boosting regressor best explains the effect of exchange rate volatility for 3 month percentage change in the dollar amount of import volume for ISIC3= ‘Başka yerde sınıflandırılmamış makine ve teçhizat’

Average Performance Metrics of Regression Algorithms

When we summarize the average results for the 6 regression models for import and export, we see that Gradient Boosting algorithm performs better than the rest of them on average for both import and export as seen in the table below.

In contrast, Decision Tree regressor seems to perform the worst among others since decision tree algorithms have a tendency to overfitting. The figures in the table below show us that Decision Tree has the highest R2 score on train with zero error for both import and export. Outperformance of decision tree verifies the overfitting problem in this case.

Average Scores of Regression Algorithms for EXPORT				
Algorithm	AVERAGE of MSE_Test	AVERAGE of R2_Test	AVERAGE of MSE_Train	AVERAGE of R2_Train
decisiontree	0.345	-0.675	0.000	0.918
gradientboosting	**0.181**	**0.082**	0.141	0.252
lasso	0.196	0.012	0.193	0.015
linear	0.195	0.018	0.190	0.030
randomforest	0.197	0.013	0.059	0.634
ridge	0.194	0.019	0.191	0.026
Grand Total	**0.218**	**-0.088**	**0.129**	**0.312**

Average Scores of Regression Algorithms for IMPORT				
Algorithm	AVERAGE of MSE_Test	AVERAGE of R2_Test	AVERAGE of MSE_Train	AVERAGE of R2_Train
decisiontree	0.390	-0.763	0.000	0.976
gradientboosting	**0.206**	**0.063**	0.147	0.307
lasso	0.216	0.011	0.214	0.016
linear	0.216	0.011	0.210	0.035
randomforest	0.223	-0.016	0.064	0.684
ridge	0.215	0.017	0.211	0.030
Grand Total	**0.244**	**-0.113**	**0.141**	**0.341**

Average R2 Test Scores of Regression Algorithms on The Three Target Features

When average R2 test scores are analyzed for the three different target features we see that;

For export:

- Linear Regression performs better for Dollar_Amount_PCT_3 with average R2 test score (0.026)
- Gradient Boosting has the highest average R2 test score (0.101) for Dollar_Amount_PCT_6
- Random Forest has also the highest average R2 test score (0.07) for Dollar_Amount_PCT_6

For import:

- Linear Regression performs better for Dollar_Amount_PCT_6 with average R2 test score (0.024)
- Gradient Boosting has highest R2 test score (0.102) on average for Dollar_Amount_PCT_6

- Random Forest has the highest R2 test score (0.060) on average for Dollar_Amount_PCT_6

Average R2 Scores on Test for EXPORT			
AVERAGE of R2_Test	*Target*		
Algorithm	DOLLAR_AMOUNT_PCT_1	DOLLAR_AMOUNT_PCT_3	DOLLAR_AMOUNT_PCT_6
decisiontree	0.000	0.000	0.000
gradientboosting	0.078	0.093	**0.101**
lasso	0.015	0.017	0.016
linear	0.022	**0.026**	0.025
randomforest	0.034	0.063	**0.070**
ridge	0.021	0.024	0.024

Average R2 Scores on Test for IMPORT			
AVERAGE of R2_Test	*Target*		
Algorithm	DOLLAR_AMOUNT_PCT_1	DOLLAR_AMOUNT_PCT_3	DOLLAR_AMOUNT_PCT_6
decisiontree	0.000	0.000	0.000
gradientboosting	0.088	0.101	**0.102**
lasso	0.014	0.016	0.016
linear	0.017	0.019	**0.024**
randomforest	0.042	0.058	**0.060**
ridge	0.019	0.022	0.024

Average R2 Test Scores for Export on BEC Breakdown

When the analysis is splitted on sectoral breakdown and product group, we see that Gradient Boosting algorithm outperforms the others in terms of average R2 test score which is greater than 5% for all of the three target feature for export.

Gradient Boosting algorithm explains on average 14% of the change in the export volume in BEC categories 'Dayanıklı tüketim malları' and 'Esası yiyecek ve içecek olan işlenmemiş tüketim malları' when all of the target features are taken into account.

Random Forest also performs similar to Gradient Boosting in such a way that it explains 13% of the change in the export volume in BEC category called 'Esası yiyecek ve içecek olan işlenmemiş tüketim malları' when all of the target features are taken into account.

Average R2 Test Scores for Export on BEC Breakdown							
AVERAGE of R2_Test		Algorithm					
Breakdown_Type	Breakdown_Name	decisiontree	gradientboosting	lasso	linear	randomforest	ridge
BEC	Başka yerde belirtilmeyen diğer mallar		**0.09**	0.02	0.02	0.05	0.02
	Binek otomobilleri		**0.12**	0.02	0.02	0.05	0.02
	Dayanıklı tüketim malları		**0.14**	0.01	0.02	**0.08**	0.01
	Dayanıksız tüketim malları		**0.12**	0.03	0.03	**0.09**	0.03
	Esası yiyecek ve içecek olan işlenmemiş hammadeler		0.04	0.02	**0.06**		0.02
	Esası yiyecek ve içecek olan işlenmemiş tüketim malları		**0.14**	0.03	0.04	**0.13**	0.04
	Esası yiyecek ve içecek olan işlenmiş tüketim malları		**0.11**	0.02	**0.06**	0.04	**0.06**
	İşlem görmemiş yakıt ve yağlar	0.00	0.00	0.00	0.00	0.00	0.00
	İşlem görmüş diğer yakıt ve yağlar		**0.08**	0.02	0.03	0.05	0.02
	Motor benzini ve diğer hafif yağlar	0.00	0.00	0.00	0.00	0.00	0.00
	Sanayi için işlem görmemiş hammaddeler		0.05	0.01	0.01		0.01
	Sanayi için işlem görmüş hammaddeler		**0.13**	0.01	0.01	**0.11**	0.01
	Sanayi ile ilgili taşımacılık araç ve gereçleri		**0.13**	0.01	**0.07**	0.05	**0.07**
	Sanayii ile ilgili olmayan taşıma araç ve gereçleri		0.04	0.02	**0.06**		0.05
	Taşımacılık araçlarının aksam ve parçaları		**0.12**	0.01	0.02	**0.08**	0.02
	Yarı dayanıklı tüketim malları		**0.13**	0.02	0.02	**0.10**	0.02
	Yatırım mallarının aksam ve parçaları		**0.12**	0.01	0.02	0.05	0.02

Average R2 Test Score for Export on ISIC3 Breakdown

When we analyze the outputs in terms of ISIC3 breakdown, we see that the highest R2 test score is again obtained with Gradient Boosting. It explains 15% of the variance in the export volume of 'Motorlu kara taşıtı ve römorklar' when all of the target features are considered.

Random Forest algorithm seems to explain 13% of that in the export volume of 'Tekstil ürünleri' when regression function is run for all of the target features.

Average R2 Test Scores for Export on ISIC3 Breakdown							
AVERAGE of R2_Test		Algorithm					
Breakdown_Type	Breakdown_Name	decisiontree	gradientboosting	lasso	linear	randomforest	ridge
ISIC3	Ağaç ve mantar ürünleri (mobilya hariç); hasır vb. örülerek yapılan maddeler		**0.06**	0.04	0.05		0.05
	Ana metal sanayi		**0.10**	0.04	0.03	**0.11**	0.05
	Atık ve hurdalar		0.02	0.01	0.01		0.01
	Balıkçılık		**0.10**	0.03	**0.07**	**0.07**	**0.07**
	Basım ve yayım; plak, kaset vb.		0.03	0.01	0.01		0.01
	Başka yerde sınıflandırılmamış elektrikli mekina ve cihazlar		**0.12**	0.03	0.03	0.03	0.03
	Büro, muhasebe ve bilgi işleme makinaları		0.04	0.04	**0.04**		0.04
	Dabaklanmış deri, bavul, el çantası, saraciye ve ayakkabı		**0.06**			0.02	
	Diğer iş faliyetleri	0.00	0.00	0.00	0.00	0.00	0.00
	Diğer ulaşım araçları		**0.09**	0.04	0.05	0.04	0.05
	Eğlence, kültür ve sporla ilgili faaliyetler	0.00	0.00	0.00	0.00	0.00	0.00
	Elektrik, gaz ve su	0.00	0.00	0.00	0.00	0.00	0.00
	Giyim eşyası		**0.20**	**0.08**	**0.10**	**0.20**	**0.10**
	Gıda ürünleri ve içecek		**0.12**	0.04	**0.05**	**0.07**	0.05
	Ham petrol ve doğalgaz	0.00	0.00	0.00	0.00	0.00	0.00
	Kağıt ve kağıt ürünleri		**0.11**	0.04	0.05	0.04	0.05
	Kok kömürü, rafine edilmiş petrol ürünleri ve nükleer yakıtlar		0.05	0.02	0.02	0.03	0.02
	Maden kömürü, linyit ve turb	0.00	0.00	0.00	0.00	0.00	0.00
	Metal eşya sanayi (makine ve teçhizatı hariç)		**0.12**	0.04	0.04	**0.06**	0.04
	Metalik olmayan diğer mineral ürünler		**0.09**	0.01	0.02	0.03	0.02
	Mobilya ve başka yerde sınıflandırılmamış diğer ürünler		**0.12**	0.01	0.01	0.05	0.01
	Motorlu kara taşıtı ve römorklar		**0.15**	0.02	0.03	**0.09**	0.03
	Ormancılık ve tomrukçuluk		0.04	0.02	0.05		0.05
	Plastik ve kauçuk ürünleri		**0.12**	0.02	0.03	0.05	0.04
	Radyo, televizyon, haberleşme teçhizatı ve cihazları		**0.12**	0.05	**0.07**	**0.06**	**0.07**
	Tarım ve hayvancılık		**0.12**	0.03	0.05	**0.07**	0.05
	Taşocakçılığı ve diğer madencilik		0.05	0.00	0.00		0.00
	Tekstil ürünleri		**0.14**	0.01	0.02	**0.13**	0.02
	Tıbbi aletler; hassas optik aletler ve saat		**0.08**	0.02	0.04	0.03	0.04
	Tütün ürünleri			0.02			0.02

Average R2 Test Score for Export on COUNTRY Breakdown

When the regression function is executed on COUNTRY breakdown, we see that Gradient Boosting explains more than 10% of the variance in the export volume of countries ABD, Almanya, Avusturya, Avustralya, İngiltere, İsveç, İtalya, Kanada, Polonya, Rusya Federasyonu, Suudi Arabistan and Yunanistan. On the other hand, Random Forest performs well on Rusya Federasyonu and Yunanistan with 15% and 11% R2 test scores, respectively.

Average R2 Test Scores for Export on COUNTRY Breakdown

AVERAGE of R2_Test		*Algorithm*				
Breakdown_Type	*Breakdown_Name*	gradientboosting	lasso	linear	randomforest	ridge
COUNTRY	ABD	**0.14**		0.00	0.07	0.00
	Almanya	**0.11**	0.00	0.01	0.08	0.01
	Avustralya	**0.13**	0.01	0.02	0.02	0.02
	Avusturya	**0.11**	0.00	0.01	0.05	0.00
	Belçika	**0.10**	0.00	0.01	0.02	0.01
	Brezilya	0.02	0.00	0.01		0.02
	Çin	0.09	0.01	0.03	0.02	0.03
	Danimarka	0.09	0.00	0.01	0.02	0.01
	Fransa	**0.10**	0.00	0.00	0.05	0.00
	Hindistan	0.07	0.01	0.01		0.01
	Hırvatistan	0.07	0.00	0.01	0.05	0.01
	Hollanda	**0.10**		0.00	0.07	0.00
	Hong Kong	0.07	0.01	0.02		0.02
	İngiltere	**0.12**	0.01	0.03	0.08	0.02
	İrlanda	0.09	0.02	0.03		0.03
	İspanya	0.09	0.00	0.01	0.03	0.01
	İsveç	**0.12**	0.00	0.01	0.04	0.01
	İsviçre	**0.10**	0.00	0.01	0.04	0.00
	İtalya	**0.14**	0.00	0.02	0.08	0.02
	Japonya	0.09	0.01	0.02	0.04	0.02
	Kanada	**0.11**	0.01	0.02	0.03	0.02
	Malta	0.03	0.00	0.00		0.01
	Meksika	0.03	0.00	0.00		0.00
	Norveç	0.09	0.01	0.01		0.01
	Polonya	**0.11**		0.01	0.05	0.00
	Portekiz	0.06	0.00	0.00		0.00
	Romanya	0.13	0.01	0.03	0.05	0.02
	Rusya Federasyonu	**0.15**	0.01	0.05	**0.15**	0.04
	Suudi Arabistan	**0.12**	0.00	0.01	0.01	0.01
	Yeni Zelanda	0.02	0.02	0.01		0.00
	Yunanistan	**0.14**	0.01	0.02	**0.11**	0.02

Average R2 Test Scores for Import on BEC Breakdown

When the regression function is run on BEC breakdown for import, it is seen that average R2 test scores are higher with Gradient Boosting. With gradient boosting on BEC breakdown on import data, average R2 test score is 17% for 'Binek otomobilleri' and 'Sanayi

ile ilgili taşımacılık araç ve gereçleri'. R2 test score is 14% for 'Taşımacılık araçlarının aksam ve parçaları'.

Average R2 Test Scores for Import on BEC Breakdown						
AVERAGE of R2_Test		Algorithm				
Breakdown_Type	Breakdown_Name	gradientboosting	lasso	linear	randomforest	ridge
BEC	Başka yerde belirtilmeyen diğer mallar		0.02	0.01		0.02
	Binek otomobilleri	**0.17**	0.06	0.07	**0.12**	0.07
	Dayanıklı tüketim malları	0.09		0.00	0.03	0.00
	Dayanıksız tüketim malları	0.08	0.00	0.00	0.04	0.00
	Esası yiyecek ve içecek olan işlenmemiş hammadeler	0.02	0.01	0.02	0.02	0.03
	Esası yiyecek ve içecek olan işlenmemiş tüketim malları	0.06	0.00	0.05		0.03
	Esası yiyecek ve içecek olan işlenmiş tüketim malları	0.06	0.04	0.03	0.04	0.04
	İşlem görmemiş yakıt ve yağlar		0.03	0.00		
	İşlem görmüş diğer yakıt ve yağlar	**0.12**	0.01	0.02	0.04	0.02
	Motor benzini ve diğer hafif yağlar	0.09	0.01	0.02	0.02	0.02
	Sanayi için işlem görmemiş hammaddeler	0.04	0.00	0.01		0.01
	Sanayi için işlem görmüş hammaddeler	**0.12**	0.00	0.01	**0.10**	0.01
	Sanayi ile ilgili taşımacılık araç ve gereçleri	**0.17**	0.02	0.04	0.09	0.04
	Sanayii ile ilgili olmayan taşıma araç ve gereçleri	0.10	0.02	0.04	0.00	0.01
	Taşımacılık araçlarının aksam ve parçaları	**0.14**	0.00	0.01	0.09	0.01
	Yarı dayanıklı tüketim malları	**0.12**	0.00	0.00	0.08	0.00
	Yatırım mallarının aksam ve parçaları	**0.12**	0.00	0.01	0.06	0.01

Average R2 Test Scores for Import on ISIC3 Breakdown

The highest R2 test scores are observed for 'Motorlu kara taşıtı ve römorklar', 'Metalik olmayan diğer mineral ürünler', 'Metal eşya sanayi (makine ve teçhizatı hariç)', 'Mobilya ve başka yerde sınıflandırılmamış diğer ürünler', 'Giyim eşyası' with 17%, 15%, 13%, 13%, 13%, respectively, when the regression function is executed on ISIC3 breakdown on import.

Average R2 Test Scores for Import on ISIC3 Breakdown							
AVERAGE of R2_Test		Algorithm					
Breakdown_Type	Breakdown_Name	decisiontree	gradientboosting	lasso	linear	randomforest	ridge
ISIC3	Ağaç ve mantar ürünleri (mobilya hariç); hasır vb. örülerek yapılan maddeler		**0.14**	0.03	0.03	0.06	0.03
	Ana metal sanayi		0.10	0.00	0.02	0.05	0.01
	Atık ve hurdalar		0.03	0.01	0.01		0.01
	Balıkçılık		0.06	0.06		0.07	0.10
	Basım ve yayım; plak, kaset vb.		0.10	0.01	0.02	0.02	0.02
	Başka yerde sınıflandırılmamış elektrikli mekina ve cihazlar		**0.12**	0.00	0.01	0.04	0.01
	Büro, muhasebe ve bilgi işleme makinaları		**0.11**	0.00	0.01	0.03	0.01
	Dabaklanmış deri, bavul, el çantası, saraciye ve ayakkabı		0.06	0.02	0.02	0.07	0.02
	Diğer hizmet faaliyetleri	0.00	0.00	0.00	0.00	0.00	0.00
	Diğer iş faliyetleri			0.05			0.03
	Diğer ulaşım araçları		0.08	0.01	0.01	0.00	0.01
	Eğlence, kültür ve sporla ilgili faaliyetler			0.01	0.02		0.03
	Elektrik, gaz ve su	0.00	0.00	0.00	0.00	0.00	0.00
	Giyim eşyası		**0.13**	0.03	0.03	0.07	0.03
	Gıda ürünleri ve içecek		0.07	0.01	0.01	0.04	0.01
	Kağıt ve kağıt ürünleri		0.07	0.03	0.03	0.01	0.03
	Kok kömürü, rafine edilmiş petrol ürünleri ve nükleer yakıtlar		0.06	0.01	0.02		0.02
	Maden kömürü, linyit ve turb			0.05	0.03		0.05
	Metal cevherleri			0.01	0.00		0.04
	Metal eşya sanayi (makine ve teçhizatı hariç)		**0.13**	0.01	0.02	0.07	0.02
	Metalik olmayan diğer mineral ürünler		**0.15**	0.05	0.06	0.06	0.06
	Mobilya ve başka yerde sınıflandırılmamış diğer ürünler		**0.13**	0.00	0.00	0.04	0.00
	Motorlu kara taşıtı ve römorklar		**0.17**	0.04	0.04	0.09	0.04
	Ormancılık ve tomrukçuluk			0.00			0.00
	Plastik ve kauçuk ürünleri		**0.13**	0.01	0.02	0.05	0.02
	Radyo, televizyon, haberleşme teçhizatı ve cihazları		0.07	0.00	0.01		0.01
	Tarım ve hayvancılık		0.06	0.02	0.01		0.02
	Taşocakçılığı ve diğer madencilik		0.07	0.03	0.04	0.02	0.03
	Tekstil ürünleri		**0.12**	0.02	0.03	0.03	0.03
	Tıbbi aletler; hassas optik aletler ve saat		**0.12**	0.00	0.02	0.05	0.02
	Tütün ürünleri		0.03	0.02	0.06		0.04

Average R2 Test Scores for Import on COUNTRY Breakdown

The Gradient Boosting algorithm outperforms against all others when the regression function is executed on country breakdown for import. When all of the target features are considered, 15% of the change in the import from ABD and Fransa, 16% of that from İtalya, 14% of that from Japonya against exchange rate volatility is explained by the underlying model.

Average R2 Test Scores for Import on COUNTRY Breakdown						
AVERAGE of R2_Test		*Algorithm*				
Breakdown_Type	*Breakdown_Name*	gradientboosting	lasso	linear	randomforest	ridge
COUNTRY	ABD	0.10	0.01	0.02	0.06	0.02
	Almanya	**0.15**		0.01	0.08	0.01
	Avustralya	0.01	0.00	0.01		0.00
	Avusturya	**0.11**	0.03	0.03	0.06	0.03
	Belçika	0.09	0.00	0.01	0.03	0.01
	Brezilya	0.06	0.00	0.01	0.00	0.01
	Çin	**0.14**	0.01	0.03	0.08	0.03
	Danimarka	0.06	0.01	0.01	0.01	0.01
	Fransa	**0.15**	0.01	0.01	0.09	0.01
	Hindistan	0.06		0.01	0.01	0.01
	Hırvatistan	0.02	0.02	0.01		0.01
	Hollanda	0.09	0.01	0.01	0.03	0.01
	Hong Kong	0.07	0.01	0.01		0.02
	İngiltere	**0.12**	0.00	0.01	0.06	0.01
	İrlanda	0.05	0.00	0.00		0.00
	İspanya	**0.12**	0.01	0.02	0.04	0.01
	İsveç	0.09	0.02	0.02	0.01	0.02
	İsviçre	**0.11**	0.00	0.00	0.03	0.00
	İtalya	**0.16**		0.01	0.13	0.01
	Japonya	**0.14**	0.03	0.04	0.04	0.04
	Kanada	0.04	0.01	0.01		0.01
	Malta		0.03	0.05	0.05	0.06
	Meksika	0.10	0.03	0.04	0.01	0.04
	Norveç	0.09	0.01	0.01	0.03	0.01
	Polonya	0.09	0.00	0.01	0.00	0.01
	Portekiz	0.06	0.00	0.02		0.01
	Romanya	**0.11**	0.01	0.03	0.07	0.02
	Rusya Federasyonu	0.08		0.00		0.01
	Suudi Arabistan	0.08	0.05	0.05	0.04	0.06
	Yeni Zelanda	0.01				
	Yunanistan	0.08	0.01	0.01		0.01

Linear Regression Results

When we consider only Linear Regression results, we see that coefficients of 16 features do not differ significantly from each other both for export and import.

In order to take a general picture on linear regression results, firstly, average of coefficients are shown in a table that takes into account all regression results for 3 different target features, and secondly, maximum value of coefficients are displayed on a seperate table.

As you can see in the maximum coefficients table below, the most exploratory variables for import and export are highlighted with red. REXR_PCT_1, REXR_PCT_3,

REXR_PCT_6, CLOSE_STD, REXR_STD, CLOSE_PCT_1, CLOSE_PCT_3 features are common for export and import.

Maximum of Coefficients in Linear Regression			
Algorithm	*Values*		
linear	MAX of Coef_YEAR	0.024	0.019
	MAX of Coef_MONTH	0.030	0.028
	MAX of Coef_REXR	1.642	**4.609**
	MAX of Coef_REXR_PCT_1	**10.085**	**7.007**
	MAX of Coef_REXR_PCT_3	**3.979**	**5.376**
	MAX of Coef_REXR_PCT_6	**4.696**	**4.578**
	MAX of Coef_CLOSE_STD	**7.979**	**8.835**
	MAX of Coef_CLOSE-OPEN_STD	**3.022**	1.335
	MAX of Coef_HIGH-LOW_STD	**6.311**	1.841
	MAX of Coef_REXR_STD	**9.811**	**11.019**
	MAX of Coef_CLOSE_PCT_1	**4.616**	**6.563**
	MAX of Coef_CLOSE_PCT_3	**6.940**	**8.426**
	MAX of Coef_CLOSE_PCT_6	3.096	**3.068**
	MAX of Coef_ISIC3	0.094	0.058
	MAX of Coef_BEC	0.038	0.287
	MAX of Coef_COUNTRY	0.011	0.002
		IMPORT	EXPORT

As seen in the average coefficients table below, the average coefficients are stronger for REXR_PCT_1, REXR_PCT_3, REXR_PCT_6, CLOSE_STD, REXR_STD, CLOSE_PCT_1, CLOSE_PCT_3 and CLOSE_PCT_6 features both for import and export.

Average of Coefficients in Linear Regression			
Algorithm	*Values*		
linear	AVERAGE of Coef_YEAR	0.000	0.002
	AVERAGE of Coef_MONTH	0.002	0.001
	AVERAGE of Coef_REXR	-0.084	-0.096
	AVERAGE of Coef_REXR_PCT_1	**1.285**	**0.256**
	AVERAGE of Coef_REXR_PCT_3	**-0.458**	**-0.231**
	AVERAGE of Coef_REXR_PCT_6	**0.543**	**0.467**
	AVERAGE of Coef_CLOSE_STD	**-1.370**	**-0.823**
	AVERAGE of Coef_CLOSE-OPEN_STD	**0.105**	0.065
	AVERAGE of Coef_HIGH-LOW_STD	**-0.290**	-0.049
	AVERAGE of Coef_REXR_STD	**1.392**	**0.605**
	AVERAGE of Coef_CLOSE_PCT_1	**-1.306**	**-0.296**
	AVERAGE of Coef_CLOSE_PCT_3	**0.480**	**0.241**
	AVERAGE of Coef_CLOSE_PCT_6	**-0.568**	**-0.404**
	AVERAGE of Coef_ISIC3	0.001	-0.006
	AVERAGE of Coef_BEC	-0.007	0.000
	AVERAGE of Coef_COUNTRY	-0.001	-0.003
		IMPORT	EXPORT

8. FUTURE DEVELOPMENTS

For those three categorical features COUNTRY, ISIC3 and BEC, label encoding method is applied to transform them into numerical format. However, instead of assigning integer values to these categorical features, the average values for the underlying year can be used so that they can be turned into continuous values and get different values for each year.

The export and import datasets composed of monthly series from 2004 to 2015. One drawback in this analysis is the lack of seasonality analysis. The seasonal components can be eliminated for further studies if time range of datasets were higher. As mentioned above, some of the sample sizes for country and industry level breakdowns are too small to run a healthy regression model. Thus, as the size of the sample and the number of years increases, seasonality analysis can be conducted on those country-level and industry-level breakdowns.

The analysis in this study are made with month ranges. Regression analysis can be extended further for yearly or quarterly analysis as well.

Econometric tests such as unit root or autocorrelation tests can also be performed before executing regression models. In statistics, a “unit root test” tests whether a time series feature is non-stationary and possesses a unit root. A unit root is a stochastic trend in a time series data. If a time series data has a unit root, it shows a systematic pattern that is unpredictable. Thus, a stationary time series data has mean, variance, autocorrelation in so that they are all constant over time. On the other side, autocorrelation in time series data refers to the situation which the errors may not be independent; i.e, errors are autocorrelated. This means that each error is correlated with the error immediately before it.

In this study, regression models are executed separately on different country-level or sector-level breakdowns. However, regression models can be applied on combinations of breakdowns such as (Country = Germany and ISIC3 = 12) or (Country = Italy and BEC = 2) etc. The outputs for these regression models may be far more explanatory than the existing ones.

8. REFERENCES

Berument, H., Dincer, N. and Mustafaoglu, Z. (2013). External income shocks and Turkish exports: A sectoral analysis. Journal of Elsevier, Economic Modelling 37 (2014) 476–484

Nazlioglu, S. (2012). Exchange rate volatility and Turkish industry-level export: Panel cointegration analysis. The Journal of International Trade & Economic Development (2013) Vol. 22, 1088–1107

Demez, S., Ustaoglu, M. (2012). Exchange-Rate Volatility' s Impact on Turkey's Exports: An Empirical Analyze for 1992-2010. Procedia - Social and Behavioral Sciences 41 (2012) 168 – 176

Demirhan, E., Demirhan, B. (2015). The Dynamic Effect of Exchange-Rate Volatility on Turkish Exports: Parsimonious Error-Correction Model Approach. Panoeconomicus (2015), Vol. 62, 429-451

Tarı, R., Yıldırım, D.Ç. (2009). The Effect of Exchange Rate on Export: An Analysis for Turkey. Yönetim ve Ekonomi (2009)

Altıntaş, H., Çetin, R., Öz, Bülent (2011). The Impact of Exchange Rate Volatility on Turkish Exports: 1993-2009. South East European Journal of Economics & Business (2011) 67-77

APPENDIX A

Python Codes for Data Preprocessing

Python codes generated by using iPython Notebook can be found in the attachment.

File Name: preprocess_monthly.ipynb, Descriptive_Analysis.html

File Type: iPython Notebook, ipython notebook generated html file

APPENDIX B

Python Codes For the Entire Study

File Name: preprocess_monthly.py

File Type: Python Script File

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Jun 30 18:43:50 2018

@author: yagmuruluturktekten
"""

# Capstone Project
# Yagmur Uluturk Tekten

# Import Libraries
import pandas as pd
import numpy as np
import os

# Change path
path = os.path.expanduser("~/Documents/capstone")
os.chdir(path)

# Read true currency map
path = os.path.expanduser("~/Documents/capstone/datasets/currency_map")
os.chdir(path)
t_currency_map = pd.read_csv('t_currency_map.csv', sep=',', header=0, encoding='utf-8')

################################################################################
##############################
#####          IMPORT DATASET PREPROCESSING
################################################################################
##############################

# Read Import datasets
```

```
path = os.path.expanduser("~/Documents/capstone/datasets/imports")
os.chdir(path)
import_19892003 = pd.read_csv('1989_2003_import.csv', sep=';', header=0,
encoding='utf-8', thousands=",")
import_20042009 = pd.read_csv('2004_2009_import.csv', sep=';', header=0,
encoding='utf-8', thousands=",")
import_20102013 = pd.read_csv('2010_2013_import.csv', sep=';', header=0,
encoding='utf-8', thousands=",")
import_20142015 = pd.read_csv('2014_2015_import.csv', sep=';', header=0,
encoding='utf-8', thousands=",")
import_dfs = [import_19892003, import_20042009, import_20102013, import_20142015]

# Different dataset files may contain different string values for same code. We should check
it.
isic3 = pd.DataFrame()
bec = pd.DataFrame()
currency = pd.DataFrame()
country = pd.DataFrame()

for dataset in import_dfs:
    isic3 = isic3.append(dataset[['ISIC3_2', 'ISIC3_ADI']].drop_duplicates(subset=None,
keep='first', inplace=False), ignore_index=False, verify_integrity=False)
    bec = bec.append(dataset[['BEC', 'BEC_ADI']].drop_duplicates(subset=None,
keep='first', inplace=False), ignore_index=False, verify_integrity=False)
    currency = currency.append(dataset[['DOVIZ_KODU',
'DOVIZ_ADI']].drop_duplicates(subset=None, keep='first', inplace=False),
ignore_index=False, verify_integrity=False)
    country = country.append(dataset[['ULKE',
'ULKE_ADI']].drop_duplicates(subset=None, keep='first', inplace=False),
ignore_index=False, verify_integrity=False)
del dataset

isic3 = isic3.drop_duplicates(subset=None, keep='first',
inplace=False)['ISIC3_2'].value_counts().sort_values(ascending=False)
bec = bec.drop_duplicates(subset=None, keep='first',
inplace=False)['BEC'].value_counts().sort_values(ascending=False)
currency = currency.drop_duplicates(subset=None, keep='first',
inplace=False)['DOVIZ_KODU'].value_counts().sort_values(ascending=False)
country = country.drop_duplicates(subset=None, keep='first',
inplace=False)['ULKE'].value_counts().sort_values(ascending=False)
```

```
# Correct Kanarya Adalari & Ceuta
print(import_19892003.iloc[6194]['ULKE_ADI']) # Kanarya Adalari
import_19892003.loc[import_19892003['ULKE_ADI'] == 'Kanarya Adaları', 'ULKE_ADI']
= 'Ceuta'
print(import_19892003.iloc[6194]['ULKE_ADI']) # Ceuta
del bec, country, currency, isic3

# Merge Import Dataframes
dfM = pd.concat(import_dfs, ignore_index=True)
del import_19892003, import_20042009, import_20102013, import_20142015, import_dfs

# Correct multiple codes for same country problem
# Find countries with duplicate codes
corrector = dfM.groupby(['ULKE_ADI', 'ULKE'])['DOLAR'].count().reset_index()
corrector.drop(['DOLAR'], axis=1, inplace=True)
corrector_keys = corrector[corrector.duplicated(['ULKE_ADI'],
keep='first')]['ULKE'].tolist()
corrector_vals = corrector[corrector.duplicated(['ULKE_ADI'],
keep='last')]['ULKE'].tolist()
corrector = dict(zip(corrector_keys, corrector_vals))
dfM.replace({'ULKE':corrector}, inplace=True)
del corrector, corrector_keys, corrector_vals

# Clean Import Dataframe dfM

# Drop first column specifying rows are about Export data
dfM.drop('IHRITH', 1, inplace=True, errors='raise')

# Drop fully NAN rows and columns
dfM.dropna(axis=0, how='all', inplace=True)
dfM.dropna(axis=1, how='all', inplace=True)

# Rename columns
dfM.columns = ['MONTH', 'BEC_CODE', 'BEC_NAME', 'DOLLAR_AMOUNT',
'CURRENCY_NAME', 'CURRENCY_CODE', 'ISIC3_CODE', 'ISIC3_NAME',
'COUNTRY_CODE', 'COUNTRY_NAME', 'YEAR']

# Check NAN values
dfM.isnull().any()
"""
Returns:
```

```
MONTH           False
BEC_CODE        False
BEC_NAME        False
DOLLAR_AMOUNT   False
CURRENCY_NAME    True
CURRENCY_CODE    True
ISIC3_CODE      False
ISIC3_NAME      False
COUNTRY_CODE    False
COUNTRY_NAME    False
YEAR            False
dtype: bool

Only CURRENCY_NAME and CURRENCY_CODE columns contain NAN values.
"""

# Fill Currency Codes & Names
print(dfM['CURRENCY_CODE'].isnull().sum()) # 10 rows
print(dfM['CURRENCY_NAME'].isnull().sum()) # 56564 rows
dfM.loc[dfM['CURRENCY_NAME'].isnull(), 'CURRENCY_CODE'] = 400 # Fill NAN
currency names and codes with US Dollar
dfM['CURRENCY_NAME'].fillna(value='ABD Doları', inplace=True) # Fill NAN
currency names and codes with US Dollar
print(dfM['CURRENCY_CODE'].isnull().sum()) # 0 rows
print(dfM['CURRENCY_NAME'].isnull().sum()) # 0 rows

# Check NAN values
dfM.isnull().any()
"""
Returns:
MONTH           False
BEC_CODE        False
BEC_NAME        False
DOLLAR_AMOUNT   False
CURRENCY_NAME   False
CURRENCY_CODE   False
ISIC3_CODE      False
ISIC3_NAME      False
COUNTRY_CODE    False
COUNTRY_NAME    False
YEAR            False
```

```
dtype: bool

None of the columns contain NAN values.
"""

# Generate new dataframes for standard names for codes, so we can drop duplicate
information from original dfM
bec_names = dfM[['BEC_CODE', 'BEC_NAME']].drop_duplicates(subset=None,
keep='first', inplace=False).reset_index(level=None, drop=True, inplace=False)
currency_names = dfM[['CURRENCY_CODE',
'CURRENCY_NAME']].drop_duplicates(subset=None, keep='first',
inplace=False).reset_index(level=None, drop=True, inplace=False)
isic3_names = dfM[['ISIC3_CODE', 'ISIC3_NAME']].drop_duplicates(subset=None,
keep='first', inplace=False).reset_index(level=None, drop=True, inplace=False)
country_names = dfM[['COUNTRY_CODE',
'COUNTRY_NAME']].drop_duplicates(subset=None, keep='first',
inplace=False).reset_index(level=None, drop=True, inplace=False)

dfM.drop(['BEC_NAME', 'CURRENCY_NAME', 'ISIC3_NAME','COUNTRY_NAME'],
axis=1, inplace=True)
dfM = dfM[['YEAR', 'MONTH', 'ISIC3_CODE', 'BEC_CODE', 'COUNTRY_CODE',
'CURRENCY_CODE', 'DOLLAR_AMOUNT']]

# Select rows after year 2003
dfM = dfM[dfM['YEAR'] > 2003]

# Drop Currency Complexity by Country
dfM.drop(['CURRENCY_CODE'], inplace=True, axis=1)
dfM = dfM.groupby(['YEAR', 'MONTH', 'ISIC3_CODE', 'BEC_CODE',
'COUNTRY_CODE'])['DOLLAR_AMOUNT'].sum().reset_index()

# Add True Currency Code
dfM = pd.merge(dfM, t_currency_map[['COUNTRY_CODE','T_CURRENCY_CODE']],
on=['COUNTRY_CODE'], how='left')

# Drop rows which we don't have information for currency and/or CPI
dfM.dropna(axis=0, how='any', inplace=True)

# Change data types
print(dfM.dtypes)
```

```
dfM['T_CURRENCY_CODE'] = dfM['T_CURRENCY_CODE'].astype('int64')
currency_names['CURRENCY_CODE'] =
currency_names['CURRENCY_CODE'].astype('int64')

print(dfM.dtypes)

# Save preprocessed import dataset information into Python Object file
path = os.path.expanduser("~/Documents/capstone/datasets/pickles")
os.chdir(path)
dfM.to_pickle("dfM.pkl", compression='gzip')
bec_names.to_pickle("bec_names_m.pkl", compression='gzip')
country_names.to_pickle("country_names_m.pkl", compression='gzip')
isic3_names.to_pickle("isic3_names_m.pkl", compression='gzip')
currency_names.to_pickle("currency_names_m.pkl", compression='gzip')
del bec_names, country_names, currency_names, isic3_names, dfM




################################################################################
##############################
#####          EXPORT DATASET PREPROCESSING
################################################################################
##############################

# Change path to back
path = os.path.expanduser("~/Documents/capstone")
os.chdir(path)

# Read Export datasets
path = os.path.expanduser("~/Documents/capstone/datasets/exports")
os.chdir(path)
# export_19892003 = pd.read_csv('1989_2003_export.csv', sep=';', header=0,
encoding='utf-8', thousands=",")
export_19892003 = pd.DataFrame()
export_20042009 = pd.read_csv('2004_2009_export.csv', sep=';', header=0,
encoding='utf-8', thousands=",")
export_20102013 = pd.read_csv('2010_2013_export.csv', sep=';', header=0,
encoding='utf-8', thousands=",")
export_20142015 = pd.read_csv('2014_2015_export.csv', sep=';', header=0,
encoding='utf-8', thousands=",")
```

```
# export_dfs = [export_19892003, export_20042009, export_20102013, export_20142015]
export_dfs = [export_20042009, export_20102013, export_20142015]

# Different dataset files may contain different string values for same code. We should check it.
isic3 = pd.DataFrame()
bec = pd.DataFrame()
currency = pd.DataFrame()
country = pd.DataFrame()

for dataset in export_dfs:
    isic3 = isic3.append(dataset[['ISIC3_2', 'ISIC3_ADI']].drop_duplicates(subset=None, keep='first', inplace=False), ignore_index=False, verify_integrity=False)
    bec = bec.append(dataset[['BEC', 'BEC_ADI']].drop_duplicates(subset=None, keep='first', inplace=False), ignore_index=False, verify_integrity=False)
    currency = currency.append(dataset[['DOVIZ_KODU', 'DOVIZ_ADI']].drop_duplicates(subset=None, keep='first', inplace=False), ignore_index=False, verify_integrity=False)
    country = country.append(dataset[['ULKE', 'ULKE_ADI']].drop_duplicates(subset=None, keep='first', inplace=False), ignore_index=False, verify_integrity=False)
del dataset

isic3 = isic3.drop_duplicates(subset=None, keep='first', inplace=False)['ISIC3_2'].value_counts().sort_values(ascending=False)
bec = bec.drop_duplicates(subset=None, keep='first', inplace=False)['BEC'].value_counts().sort_values(ascending=False)
currency = currency.drop_duplicates(subset=None, keep='first', inplace=False)['DOVIZ_KODU'].value_counts().sort_values(ascending=False)
country = country.drop_duplicates(subset=None, keep='first', inplace=False)['ULKE'].value_counts().sort_values(ascending=False)

"""
print(export_19892003.iloc[6194]['ULKE_ADI']) # Kanarya Adalari
export_19892003.loc[export_19892003['ULKE_ADI'] == 'Kanarya Adaları', 'ULKE_ADI'] = 'Ceuta'
print(export_19892003.iloc[6194]['ULKE_ADI']) # Ceuta
"""
del bec, country, currency, isic3
```

```
# Merge Export Dataframes
dfX = pd.concat(export_dfs, ignore_index=True)
del export_19892003,  export_20042009, export_20102013, export_20142015, export_dfs

"""
# Correct multiple codes for same country problem
corrector = dfX.groupby(['ULKE_ADI', 'ULKE'])['DOLAR'].count().reset_index()
corrector.drop(['DOLAR'], axis=1, inplace=True)
corrector_keys = corrector[corrector.duplicated(['ULKE_ADI'],
keep='first')]['ULKE'].tolist()
corrector_vals = corrector[corrector.duplicated(['ULKE_ADI'],
keep='last')]['ULKE'].tolist()
corrector = dict(zip(corrector_keys, corrector_vals))
dfX.replace({'ULKE':corrector}, inplace=True)
del corrector, corrector_keys, corrector_vals
# Important note: Duplicate country code problem does not exist in dfX dataframe
"""

# Correct country code of Switzerland
dfX["ULKE"].replace(39, 36, inplace=True)

# Clean Export Dataframe dfX

# Drop first column specifying rows are about Export data
dfX.drop('IHRITH', 1, inplace=True, errors='raise')

# Drop fully NAN rows and columns
dfX.dropna(axis=0, how='all', inplace=True)
dfX.dropna(axis=1, how='all', inplace=True)

# Rename columns
dfX.columns = ['MONTH', 'BEC_CODE', 'BEC_NAME', 'DOLLAR_AMOUNT',
'CURRENCY_NAME', 'CURRENCY_CODE', 'ISIC3_CODE', 'ISIC3_NAME',
'COUNTRY_CODE', 'COUNTRY_NAME', 'YEAR']

# Check NAN values
dfX.isnull().any()
"""
Returns:
MONTH          False
BEC_CODE       False
```

```
BEC_NAME        False
DOLLAR_AMOUNT    False
CURRENCY_NAME    False
CURRENCY_CODE    False
ISIC3_CODE       False
ISIC3_NAME       False
COUNTRY_CODE     False
COUNTRY_NAME     False
YEAR             False
dtype: bool

None of the columns contains NAN values.
"""


# Generate new dataframes for standard names for codes, so we can drop duplicate
information from original dfX
bec_names = dfX[['BEC_CODE', 'BEC_NAME']].drop_duplicates(subset=None,
keep='first', inplace=False).reset_index(level=None, drop=True, inplace=False)
currency_names = dfX[['CURRENCY_CODE',
'CURRENCY_NAME']].drop_duplicates(subset=None, keep='first',
inplace=False).reset_index(level=None, drop=True, inplace=False)
isic3_names = dfX[['ISIC3_CODE', 'ISIC3_NAME']].drop_duplicates(subset=None,
keep='first', inplace=False).reset_index(level=None, drop=True, inplace=False)
country_names = dfX[['COUNTRY_CODE',
'COUNTRY_NAME']].drop_duplicates(subset=None, keep='first',
inplace=False).reset_index(level=None, drop=True, inplace=False)

dfX.drop(['BEC_NAME', 'CURRENCY_NAME', 'ISIC3_NAME','COUNTRY_NAME'],
axis=1, inplace=True)
dfX = dfX[['YEAR', 'MONTH', 'ISIC3_CODE', 'BEC_CODE', 'COUNTRY_CODE',
'CURRENCY_CODE', 'DOLLAR_AMOUNT']]


# Drop Currency Complexity by Country
dfX.drop(['CURRENCY_CODE'], inplace=True, axis=1)
dfX = dfX.groupby(['YEAR', 'MONTH', 'ISIC3_CODE', 'BEC_CODE',
'COUNTRY_CODE'])['DOLLAR_AMOUNT'].sum().reset_index()

# Add True Currency Code
dfX = pd.merge(dfX, t_currency_map[['COUNTRY_CODE','T_CURRENCY_CODE']],
```

```
on=['COUNTRY_CODE'], how='left')

# Drop rows which we don't have information for currency and/or CPI
dfX.dropna(axis=0, how='any', inplace=True)

# Change data types
print(dfX.dtypes)

dfX['T_CURRENCY_CODE'] = dfX['T_CURRENCY_CODE'].astype('int64')
currency_names['CURRENCY_CODE'] =
currency_names['CURRENCY_CODE'].astype('int64')

print(dfX.dtypes)

# Save preprocessed export dataset information into Python Object file
path = os.path.expanduser("~/Documents/capstone/datasets/pickles")
os.chdir(path)
dfX.to_pickle("dfX.pkl", compression='gzip')
bec_names.to_pickle("bec_names_x.pkl", compression='gzip')
country_names.to_pickle("country_names_x.pkl", compression='gzip')
isic3_names.to_pickle("isic3_names_x.pkl", compression='gzip')
currency_names.to_pickle("currency_names_x.pkl", compression='gzip')
del bec_names, country_names, currency_names, isic3_names, dfX

# Change path to back
path = os.path.expanduser("~/Documents/capstone")
os.chdir(path)
```

File Name: cpi.py

File Type: Python Script File

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Jul 22 18:40:30 2018

@author: yagmuruluturktekten
"""

# Capstone Project
# Yagmur Uluturk Tekten

################################################################################
################################
#####          CPI VALUES TO EXPORT & IMPORT DATAFRAMES
################################################################################
################################

# Import Libraries
import pandas as pd
import numpy as np
import os

# Change path
path = os.path.expanduser("~/Documents/capstone")
os.chdir(path)

# Load CPI csv file

# Change path firstly
path = os.path.expanduser("~/Documents/capstone/datasets/cpi")
os.chdir(path)

# Read CPI Indices CSV filer
cpi = pd.read_csv('cpi.csv', sep=',', header=0, encoding='utf-8')
cpi.drop(['MONTH','YEAR'], axis=1, inplace=True, errors='raise')
```

```
# Melt CPI df
cpi_cols = list(cpi.columns)[2:]
cpi = pd.melt(cpi, id_vars = ['MONTH&YEAR'], value_vars = cpi_cols,
var_name='COUNTRY_NAME', value_name='CPI')

# Generate month & year separately
cpi['DATE_PD'] = pd.to_datetime(cpi['MONTH&YEAR'], format = '%d.%m.%Y')
cpi['MONTH'] = cpi['DATE_PD'].map(lambda x: x.month)
cpi['YEAR'] = cpi['DATE_PD'].map(lambda x: x.year)

# Reshape CPI df
cpi = cpi[['YEAR', 'MONTH', 'DATE_PD', 'COUNTRY_NAME', 'CPI']]

# CPI of Turkey
cpi_tr = cpi[cpi['COUNTRY_NAME'] == 'Turkey']
cpi_tr = cpi_tr[['YEAR','MONTH','CPI']]
cpi_tr.rename(columns={'CPI':'CPI_TR'}, inplace = True)

# Map country names
country_names_eng_tur = {   'United States':'ABD',
                        'Germany':'Almanya',
                        'Australia':'Avustralya',
                        'Austria':'Avusturya',
                        'Belgium':'Belçika',
                        'Brazil':'Brezilya',
                        'Czech Republic':'Çek Cumhuriyeti',
                        'China':'Çin',
                        'Denmark':'Danimarka',
                        'France':'Fransa',
                        'South Africa':'Güney Afrika',
                        'India':'Hindistan',
                        'Croatia':'Hırvatistan',
                        'Netherlands':'Hollanda',
                        'Hong Kong SAR':'Hong Kong',
                        'United Kingdom':'İngiltere',
                        'Ireland':'İrlanda',
                        'Spain':'İspanya',
                        'Israel':'İsrail',
                        'Sweden':'İsveç',
                        'Switzerland':'İsviçre',
                        'Italy':'İtalya',
```

```
                    'Japan':'Japonya',
                    'Canada':'Kanada',
                    'Mexico':'Meksika',
                    'Norway':'Norveç',
                    'Poland':'Polonya',
                    'Portugal':'Portekiz',
                    'Romania':'Romanya',
                    'Russia':'Rusya Federasyonu',
                    'Saudi Arabia':'Suudi Arabistan',
                    'New Zealand':'Yeni Zelanda',
                    'Greece':'Yunanistan'
        }

cpi.replace({"COUNTRY_NAME": country_names_eng_tur}, inplace=True)

# Drop unwanted countries
countries_to_keep = list(country_names_eng_tur.values())
countries_to_keep.append('Malta')
cpi = cpi[cpi['COUNTRY_NAME'].isin(countries_to_keep)]

# Find Country Codes
path = os.path.expanduser("~/Documents/capstone/datasets/pickles")
os.chdir(path)
country_names_m = pd.read_pickle('country_names_m.pkl', compression='gzip')
cpi = pd.merge(cpi, country_names_m, on=['COUNTRY_NAME'], how='left')

# Drop columns & reorder
cpi = cpi[['YEAR', 'MONTH', 'COUNTRY_CODE', 'CPI']]

# Add CPI value to dfM & dfX
dfX = pd.read_pickle('dfX.pkl', compression = 'gzip')
dfM = pd.read_pickle('dfM.pkl', compression = 'gzip')

dfX = pd.merge(dfX, cpi, on=['YEAR','MONTH','COUNTRY_CODE'], how='left')
dfM = pd.merge(dfM, cpi, on=['YEAR','MONTH','COUNTRY_CODE'], how='left')

# Add CPI Turkey to dfM & dfX
dfX = pd.merge(dfX, cpi_tr, on=['YEAR','MONTH'], how='left')
dfM = pd.merge(dfM, cpi_tr, on=['YEAR','MONTH'], how='left')

# Save CPI values into Python Object file
```

```
path = os.path.expanduser("~/Documents/capstone/datasets/pickles")
os.chdir(path)
dfX.to_pickle("dfX_wcpi.pkl", compression='gzip')
dfM.to_pickle("dfM_wcpi.pkl", compression='gzip')
del countries_to_keep, country_names_eng_tur, cpi_cols

# Change path to back
path = os.path.expanduser("~/Documents/capstone")
os.chdir(path)
```

File Name: currency_preprocess.py

File Type: Python Script File

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Jul 15 15:35:06 2018

@author: yagmuruluturktekten
"""

# Capstone Project
# Yagmur Uluturk Tekten

################################################################################
################################
#####          CURRENCY PAIRS PREPROCESSING
################################################################################
################################

# Import Libraries
import pandas as pd
import numpy as np
import os
import glob

# Change path
path = os.path.expanduser("~/Documents/capstone")
os.chdir(path)

dfX = pd.read_pickle('datasets/pickles/dfX_wcpi.pkl', compression = 'gzip')
dfM = pd.read_pickle('datasets/pickles/dfM_wcpi.pkl', compression = 'gzip')

# Read true currency map
path = os.path.expanduser("~/Documents/capstone/datasets/currency_map")
os.chdir(path)
t_currency_map = pd.read_csv('t_currency_map.csv', sep=',', header=0, encoding='utf-8')

# Load currency csv files
```

```
# Change path firstly
path = os.path.expanduser("~/Documents/capstone/datasets/currencies")
os.chdir(path)

# Read all currency CSV file
files = glob.glob(path + '/*.csv')
currency_dfs = []

for csv_file in files:
    df = pd.read_csv(csv_file, sep=',', header=0, encoding='utf-8', thousands=",")
    df.name = csv_file[66:73].lower().replace('_', '')
    currency_dfs.append(df)
del df, csv_file

# Drop Change % column for each currency df and add currency code as a column
new_currency_dfs = []

for currency in currency_dfs:
    currency.drop(['Change %'], axis=1, inplace = True)
    currency['CURRENCY_PAIR'] = currency.name
    currency = pd.merge(currency,
t_currency_map[['CURRENCY_PAIR','T_CURRENCY_CODE']],
on=['CURRENCY_PAIR'], how='left')
    currency.drop(['CURRENCY_PAIR'], axis=1, inplace = True)
    new_currency_dfs.append(currency)
del currency

currency_dfs = new_currency_dfs
del new_currency_dfs

# Merge currency data frames and create Year and Month columns seperately
currency_df = pd.concat(currency_dfs, ignore_index=True)
del currency_dfs
currency_df['DATE_PD'] = pd.to_datetime(currency_df['Date'], format = '%b %y')
currency_df['MONTH'] = currency_df['DATE_PD'].map(lambda x: x.month)
currency_df['YEAR'] = currency_df['DATE_PD'].map(lambda x: x.year)

# Change path
path = os.path.expanduser("~/Documents/capstone")
os.chdir(path)
```

```
# Drop duplicate rows
currency_df.drop_duplicates(subset=['T_CURRENCY_CODE', 'MONTH','YEAR'],
keep='first', inplace=True)

# Join dfM & dfX with currency dataframe
dfM = pd.merge(dfM,
currency_df[['YEAR','MONTH','T_CURRENCY_CODE','Price','Open','High','Low']],
on=['YEAR','MONTH','T_CURRENCY_CODE'], how='left')
dfM = dfM.rename(columns={'Price': 'CLOSE', 'Open':'OPEN', 'High':'HIGH',
'Low':'LOW'})
dfX = pd.merge(dfX,
currency_df[['YEAR','MONTH','T_CURRENCY_CODE','Price','Open','High','Low']],
on=['YEAR','MONTH','T_CURRENCY_CODE'], how='left')
dfX = dfX.rename(columns={'Price': 'CLOSE', 'Open':'OPEN', 'High':'HIGH',
'Low':'LOW'})

# REXR, CLOSE-OPEN, HIGH-LOW
dfM['REXR'] = dfM['CLOSE'] * (dfM['CPI'] / dfM['CPI_TR'])
dfM['CLOSE-OPEN'] = dfM['CLOSE'] - dfM['OPEN']
dfM['HIGH-LOW'] = dfM['HIGH'] - dfM['LOW']

dfX['REXR'] = dfX['CLOSE'] * (dfX['CPI'] / dfX['CPI_TR'])
dfX['CLOSE-OPEN'] = dfX['CLOSE'] - dfX['OPEN']
dfX['HIGH-LOW'] = dfX['HIGH'] - dfX['LOW']

# Drop NA, if exists
dfM.dropna(axis=0, how='any', inplace=True)
dfX.dropna(axis=0, how='any', inplace=True)

# Convert dfM & dfX to Pandas time-series format
dfM['MONTH_YEAR'] = dfM['MONTH'].map(str) + '/' + dfM['YEAR'].map(str)
dfM['MONTH_YEAR'] = pd.to_datetime(dfM['MONTH_YEAR'], errors='raise',
yearfirst=True, format='%m/%Y').dt.to_period('M')
dfM['SUPER_INDEX'] = dfM['COUNTRY_CODE'].map(str) + '/' +
dfM['ISIC3_CODE'].map(str) + '/' + dfM['BEC_CODE'].map(str)
dfM.set_index(['MONTH_YEAR','SUPER_INDEX'], drop=False, append=False,
inplace=True, verify_integrity=False)
dfM.sort_index(axis=0, level=[0,1], ascending=[True,True], inplace=True,
sort_remaining=False)
```

```
dfX['MONTH_YEAR'] = dfX['MONTH'].map(str) + '/' + dfX['YEAR'].map(str)
dfX['MONTH_YEAR'] = pd.to_datetime(dfX['MONTH_YEAR'], errors='raise',
yearfirst=True, format='%m/%Y').dt.to_period('M')
dfX['SUPER_INDEX'] = dfX['COUNTRY_CODE'].map(str) + '/' +
dfX['ISIC3_CODE'].map(str) + '/' + dfX['BEC_CODE'].map(str)
dfX.set_index(['MONTH_YEAR','SUPER_INDEX'], drop=False, append=False,
inplace=True, verify_integrity=False)
dfX.sort_index(axis=0, level=[0,1], ascending=[True,True], inplace=True,
sort_remaining=False)


# Rolling variables & functions
list_df = [dfM, dfX]
list_stds = ['CLOSE','CLOSE-OPEN','HIGH-LOW','REXR']
months = [[6,3],]
del dfM, dfX

def rolling_stones(df, col, month, min_month):
    std_df = df.copy()
    std_df = std_df[col].unstack()
    std_df = std_df.pct_change(periods=1, fill_method=None, limit=None)
    std_df = std_df.rolling(month, min_periods=min_month).std()
    std_df = std_df.stack().reset_index().rename(columns={0:col+'_STD'})
    df = pd.merge(df, std_df,  on=['MONTH_YEAR','SUPER_INDEX'], how='left')

    df.set_index(['MONTH_YEAR','SUPER_INDEX'], drop=False, append=False,
inplace=True, verify_integrity=False)
    df.sort_index(axis=0, level=[0,1], ascending=[True,True], inplace=True,
sort_remaining=False)

    del std_df
    return df

for i in range(0,len(list_df)):
    for feature in list_stds:
        for time_range in months:
            list_df[i] = rolling_stones(list_df[i], feature, *time_range)

dfM = list_df[0]
dfX = list_df[1]
del feature, i, list_df, list_stds, months
```

```
# Save preprocessed import & export dataset information into Python Object file
path = os.path.expanduser("~/Documents/capstone/datasets/pickles")
os.chdir(path)
dfM.to_pickle('dfM_wfeatures.pkl', compression='gzip')
dfX.to_pickle('dfX_wfeatures.pkl', compression='gzip')
currency_df.to_pickle('currency.pkl', compression='gzip')

# Change path to back
path = os.path.expanduser("~/Documents/capstone")
os.chdir(path)
```

File Name: change_calculator.py

File Type: Python Script File

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Aug 22 15:21:40 2018

@author: yagmuruluturktekten
"""

# Capstone Project
# Yagmur Uluturk Tekten

################################################################################
##############################
#####          CHANGE CALCULATOR
################################################################################
##############################

# Import Libraries
import pandas as pd
import numpy as np
import os

# Change path
path = os.path.expanduser("~/Documents/capstone")
os.chdir(path)

# Read pickles for Import and Export dataframes containing features
dfM_withfeatures = pd.read_pickle('datasets/pickles/dfM_wfeatures.pkl', compression =
'gzip')
dfX_withfeatures = pd.read_pickle('datasets/pickles/dfX_wfeatures.pkl', compression =
'gzip')



# Percentage change variables & functions
list_df = [dfM_withfeatures, dfX_withfeatures]
list_pct = ['DOLLAR_AMOUNT', 'REXR', 'CLOSE']
windows = [1,3,6]
del dfM_withfeatures, dfX_withfeatures
```

```
def rolling_stones(df, col, window):
    pct_df = df.copy()
    pct_df = pct_df[col].unstack()
    pct_df = pct_df.pct_change(periods = window, fill_method='pad', limit=None)
    pct_df = pct_df.stack().reset_index().rename(columns={0:col+'_PCT_'+ str(window)})
    df = pd.merge(df, pct_df,  on=['MONTH_YEAR','SUPER_INDEX'], how='left')
    df.set_index(['MONTH_YEAR','SUPER_INDEX'], drop=False, append=False,
inplace=True, verify_integrity=False)
    df.sort_index(axis=0, level=[0,1], ascending=[True,True], inplace=True,
sort_remaining=False)

    del pct_df
    return df

for i in range(0,len(list_df)):
    for feature in list_pct:
        for window in windows:
            list_df[i] = rolling_stones(list_df[i], feature, window)

dfM = list_df[0]
dfX = list_df[1]
del feature, i, list_df, list_pct, window

# Save preprocessed import & export dataset information into Python Object file
path = os.path.expanduser("~/Documents/capstone/datasets/pickles")
os.chdir(path)
dfM.to_pickle('dfM_wfeatures_pct.pkl', compression='gzip')
dfX.to_pickle('dfX_wfeatures_pct.pkl', compression='gzip')
```

File Name: model.py

File Type: Python Script File

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jul 23 23:41:28 2018

@author: yagmuruluturktekten
"""

# Change path
import os
path = os.path.expanduser("~/Documents/capstone")
os.chdir(path)

# Read pickles for Import and Export dataframes containing features

from datetime import datetime
start_time = datetime.now()

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import MinMaxScaler

from sklearn import linear_model
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor

dfM_withfeatures = pd.read_pickle('datasets/pickles/dfM_wfeatures_pct.pkl', compression
= 'gzip')
dfX_withfeatures = pd.read_pickle('datasets/pickles/dfX_wfeatures_pct.pkl', compression =
'gzip')

# Handle Missing Values in dfM_withfeatures
dfM_withfeatures.isnull().sum()

"""
There are 13 columns which have nan values
CLOSE_STD             46751
CLOSE-OPEN_STD        51262
```

```
HIGH-LOW_STD          51173
REXR_STD             46751
DOLLAR_AMOUNT_PCT_1     3923
DOLLAR_AMOUNT_PCT_3     8527
DOLLAR_AMOUNT_PCT_6    15428
REXR_PCT_1            3923
REXR_PCT_3            8527
REXR_PCT_6           15428
CLOSE_PCT_1           3923
CLOSE_PCT_3           8527
CLOSE_PCT_6          15428
"""

# Drop rows with nan values
dfM_withfeatures.dropna(inplace=True)


# Handle Missing Values in dfX_withfeatures
dfX_withfeatures.isnull().sum()

"""
There are 13 columns which have nan values
CLOSE_STD             46705
CLOSE-OPEN_STD        50581
HIGH-LOW_STD          50435
REXR_STD             46705
DOLLAR_AMOUNT_PCT_1     3872
DOLLAR_AMOUNT_PCT_3     8235
DOLLAR_AMOUNT_PCT_6    14760
REXR_PCT_1            3872
REXR_PCT_3            8235
REXR_PCT_6           14760
CLOSE_PCT_1           3872
CLOSE_PCT_3           8235
CLOSE_PCT_6          14760
"""

# Drop rows with nan values
dfX_withfeatures.dropna(inplace=True)

# Select necessary columns for modelling

dfM_model = dfM_withfeatures[['YEAR', 'MONTH', 'ISIC3_CODE', 'BEC_CODE',
'COUNTRY_CODE','REXR', 'REXR_PCT_1', 'REXR_PCT_3', 'REXR_PCT_6',
'CLOSE_STD', 'CLOSE-OPEN_STD', 'HIGH-LOW_STD', 'REXR_STD',
'CLOSE_PCT_1', 'CLOSE_PCT_3', 'CLOSE_PCT_6', 'DOLLAR_AMOUNT',
```

```
'DOLLAR_AMOUNT_PCT_1', 'DOLLAR_AMOUNT_PCT_3',
'DOLLAR_AMOUNT_PCT_6']]
dfX_model = dfX_withfeatures[['YEAR', 'MONTH', 'ISIC3_CODE', 'BEC_CODE',
'COUNTRY_CODE','REXR', 'REXR_PCT_1', 'REXR_PCT_3', 'REXR_PCT_6',
'CLOSE_STD', 'CLOSE-OPEN_STD', 'HIGH-LOW_STD', 'REXR_STD',
'CLOSE_PCT_1', 'CLOSE_PCT_3', 'CLOSE_PCT_6', 'DOLLAR_AMOUNT',
'DOLLAR_AMOUNT_PCT_1', 'DOLLAR_AMOUNT_PCT_3',
'DOLLAR_AMOUNT_PCT_6']]




# Apply LABEL ENCODING for categorical variables in Import dataframe using
"cat.codes" method !!!
# There are 3 categorical variables(ISIC3_CODE,BEC_CODE,COUNTRY_CODE) that
need to be transformed into numbers in order to use them in the modelling
# Label encoding simply converts each value in the column to a number that stands for the
index of the column

# Label enconding for ISIC3_CODE in Import data frame
dfM_model.ISIC3_CODE = pd.Categorical(dfM_model.ISIC3_CODE)
dfM_model['ISIC3'] = dfM_model.ISIC3_CODE.cat.codes
dfM_model['ISIC3'].value_counts().sort_index()

# Label enconding for ISIC3_CODE in Export data frame
dfX_model.ISIC3_CODE = pd.Categorical(dfX_model.ISIC3_CODE)
dfX_model['ISIC3'] = dfX_model.ISIC3_CODE.cat.codes
dfX_model['ISIC3'].value_counts().sort_index()

# Label enconding for BEC_CODE in Import data frame
dfM_model.BEC_CODE = pd.Categorical(dfM_model.BEC_CODE)
dfM_model['BEC'] = dfM_model.BEC_CODE.cat.codes
dfM_model['BEC'].value_counts().sort_index()

# Label enconding for BEC_CODE in Export data frame
dfX_model.BEC_CODE = pd.Categorical(dfX_model.BEC_CODE)
dfX_model['BEC'] = dfX_model.BEC_CODE.cat.codes
dfX_model['BEC'].value_counts().sort_index()

# Label enconding for COUNTRY_CODE in Import data frame
dfM_model.COUNTRY_CODE = pd.Categorical(dfM_model.COUNTRY_CODE)
dfM_model['COUNTRY'] = dfM_model.COUNTRY_CODE.cat.codes
dfM_model['COUNTRY'].value_counts().sort_index()

# Label enconding for COUNTRY_CODE in Export data frame
```

```
dfX_model.COUNTRY_CODE = pd.Categorical(dfX_model.COUNTRY_CODE)
dfX_model['COUNTRY'] = dfX_model.COUNTRY_CODE.cat.codes
dfX_model['COUNTRY'].value_counts().sort_index()


dfM_model.dtypes
dfX_model.dtypes


# Convert ISIC3_CODE, BEC_CODE and COUNTRY_CODE which is in categorical
format into numerical format
dfM_model['ISIC3_CODE'] = dfM_model['ISIC3_CODE'].astype('int64', copy=True)
dfM_model['BEC_CODE'] = dfM_model['BEC_CODE'].astype('int64', copy=True)
dfM_model['COUNTRY_CODE'] = dfM_model['COUNTRY_CODE'].astype('int64',
copy=True)
dfM_model.dtypes


dfX_model['ISIC3_CODE'] = dfX_model['ISIC3_CODE'].astype('int64', copy=True)
dfX_model['BEC_CODE'] = dfX_model['BEC_CODE'].astype('int64', copy=True)
dfX_model['COUNTRY_CODE'] = dfX_model['COUNTRY_CODE'].astype('int64',
copy=True)
dfX_model.dtypes

# Keep original ISIC3_CODE,COUNTRY_CODE and BEC_CODE to remember the
names of the sectors and countries for further analysis

# For Import
isic3_code_m = dfM_model[['ISIC3_CODE','ISIC3']]
isic3_code_m = isic3_code_m.drop_duplicates(subset=None, keep='first',
inplace=False).reset_index(level=None, drop=True, inplace=False)

bec_code_m = dfM_model[['BEC_CODE','BEC']]
bec_code_m = bec_code_m.drop_duplicates(subset=None, keep='first',
inplace=False).reset_index(level=None, drop=True, inplace=False)

country_code_m = dfM_model[['COUNTRY_CODE','COUNTRY']]
country_code_m = country_code_m.drop_duplicates(subset=None, keep='first',
inplace=False).reset_index(level=None, drop=True, inplace=False)

# For Export
isic3_code_x = dfX_model[['ISIC3_CODE','ISIC3']]
isic3_code_x = isic3_code_x.drop_duplicates(subset=None, keep='first',
inplace=False).reset_index(level=None, drop=True, inplace=False)

bec_code_x = dfX_model[['BEC_CODE','BEC']]
```

```
bec_code_x = bec_code_x.drop_duplicates(subset=None, keep='first',
inplace=False).reset_index(level=None, drop=True, inplace=False)

country_code_x = dfX_model[['COUNTRY_CODE','COUNTRY']]
country_code_x = country_code_x.drop_duplicates(subset=None, keep='first',
inplace=False).reset_index(level=None, drop=True, inplace=False)


# Read pickles for ISIC3, COUNTRY and BEC names
isic3_names_m = pd.read_pickle('datasets/pickles/isic3_names_m.pkl', compression =
'gzip')
isic3_names_x = pd.read_pickle('datasets/pickles/isic3_names_x.pkl', compression = 'gzip')

bec_names_m = pd.read_pickle('datasets/pickles/bec_names_m.pkl', compression = 'gzip')
bec_names_x = pd.read_pickle('datasets/pickles/bec_names_x.pkl', compression = 'gzip')

country_names_m = pd.read_pickle('datasets/pickles/country_names_m.pkl', compression
= 'gzip')
country_names_x = pd.read_pickle('datasets/pickles/country_names_x.pkl', compression =
'gzip')


# Combine codes and names for ISIC3, COUNTRY and BEC

isic3_map_m = pd.merge(isic3_code_m, isic3_names_m, on=['ISIC3_CODE'], how='left')
isic3_map_x = pd.merge(isic3_code_x, isic3_names_x, on=['ISIC3_CODE'], how='left')

country_map_m = pd.merge(country_code_m, country_names_m,
on=['COUNTRY_CODE'], how='left')
country_map_x = pd.merge(country_code_x, country_names_x, on=['COUNTRY_CODE'],
how='left')

bec_map_m = pd.merge(bec_code_m, bec_names_m, on=['BEC_CODE'], how='left')
bec_map_x = pd.merge(bec_code_x, bec_names_x, on=['BEC_CODE'], how='left')


# Save mapping files as pickles

# Change path first
path = os.path.expanduser("~/Documents/capstone/datasets/pickles")
os.chdir(path)

isic3_map_m.to_pickle("isic3_map_m.pkl", compression='gzip')
isic3_map_x.to_pickle("isic3_map_x.pkl", compression='gzip')
country_map_m.to_pickle("country_map_m.pkl", compression='gzip')
country_map_x.to_pickle("country_map_x.pkl", compression='gzip')
```

```
bec_map_m.to_pickle("bec_map_m.pkl", compression='gzip')
bec_map_x.to_pickle("bec_map_x.pkl", compression='gzip')

############################################### OUTLIERS
###############################################

# Detect OUTLIERS in Import dataframe

def remove_outliers(df_man, col, threshold):
    df = df_man.copy()
    del_df = df[df[col] > threshold]
    print('There are %d outliers in the %s out of %d entries' % (len(del_df), str(col), len(df)))
    df = df[df[col] <= threshold]
    return df

dfM_model = remove_outliers(dfM_model, 'DOLLAR_AMOUNT_PCT_1', 1)
dfM_model = remove_outliers(dfM_model, 'DOLLAR_AMOUNT_PCT_3', 1)
dfM_model = remove_outliers(dfM_model, 'DOLLAR_AMOUNT_PCT_6', 1)
dfM_model = remove_outliers(dfM_model, 'REXR', 200000)
dfM_model = remove_outliers(dfM_model, 'CLOSE-OPEN_STD', 90)
dfM_model = remove_outliers(dfM_model, 'HIGH-LOW_STD', 90)
dfM_model = remove_outliers(dfM_model, 'CLOSE-OPEN_STD', 90)

dfX_model = remove_outliers(dfX_model, 'DOLLAR_AMOUNT_PCT_1', 1)
dfX_model = remove_outliers(dfX_model, 'DOLLAR_AMOUNT_PCT_3', 1)
dfX_model = remove_outliers(dfX_model, 'DOLLAR_AMOUNT_PCT_6', 1)
dfX_model = remove_outliers(dfX_model, 'REXR', 200000)
dfX_model = remove_outliers(dfX_model, 'CLOSE-OPEN_STD', 90)
dfX_model = remove_outliers(dfX_model, 'HIGH-LOW_STD', 20)
dfX_model = remove_outliers(dfX_model, 'CLOSE-OPEN_STD', 90)

############################################### REGRESSION MODEL
###############################################

# Prepare features and targets

dfM_targets =
dfM_model[['DOLLAR_AMOUNT_PCT_1','DOLLAR_AMOUNT_PCT_3','DOLLAR_A
MOUNT_PCT_6']]
dfM_targets.reset_index(drop=True, inplace=True)
dfX_targets =
dfX_model[['DOLLAR_AMOUNT_PCT_1','DOLLAR_AMOUNT_PCT_3','DOLLAR_A
MOUNT_PCT_6']]
dfX_targets.reset_index(drop=True, inplace=True)

dfM_model.drop(['DOLLAR_AMOUNT','DOLLAR_AMOUNT_PCT_1','DOLLAR_AM
```

```
OUNT_PCT_3','DOLLAR_AMOUNT_PCT_6',
            'ISIC3_CODE', 'BEC_CODE', 'COUNTRY_CODE'],
           axis=1, inplace=True)
dfX_model.drop(['DOLLAR_AMOUNT','DOLLAR_AMOUNT_PCT_1','DOLLAR_AMO
UNT_PCT_3','DOLLAR_AMOUNT_PCT_6',
            'ISIC3_CODE', 'BEC_CODE', 'COUNTRY_CODE'],
           axis=1, inplace=True)
dfM_model.reset_index(drop=True, inplace=True)
dfX_model.reset_index(drop=True, inplace=True)


# Feature Scaling

# Year
dfM_model['YEAR'] = dfM_model['YEAR'] - 2003
dfX_model['YEAR'] = dfX_model['YEAR'] - 2003
# Scaling year had very little effect on MSE. It reduced MSE slightly.


# HIGH-LOW_STD, CLOSE-OPEN_STD
scaler = MinMaxScaler()

dfM_model[['HIGH-LOW_STD', 'CLOSE-OPEN_STD']] =
scaler.fit_transform(dfM_model[['HIGH-LOW_STD', 'CLOSE-OPEN_STD']])
dfX_model[['HIGH-LOW_STD', 'CLOSE-OPEN_STD']] =
scaler.fit_transform(dfX_model[['HIGH-LOW_STD', 'CLOSE-OPEN_STD']])


# Tried: MinMaxScaler, StandardScaler, MaxAbsScaler, RobustScaler, Normalizer
# Best results are observed with MinMaxScaler


# Put features and targets in the some box
model_inputs = [[dfM_model, dfM_targets, 'import'],
            [dfX_model, dfX_targets, 'export']

            ]

##############################################################################
##

# Azure Machine Learning Datasets
dfM_azure = dfM_model.copy()
dfM_azure['DOLLAR_AMOUNT_PCT_1'] =
dfM_targets['DOLLAR_AMOUNT_PCT_1']
```

```
dfX_azure = dfX_model.copy()
dfX_azure['DOLLAR_AMOUNT_PCT_1'] = dfX_targets['DOLLAR_AMOUNT_PCT_1']

path = os.path.expanduser("~/Documents/capstone/datasets/azure")
os.chdir(path)

dfM_azure.to_csv('dfm_azure.csv',
                 sep=',',
                 na_rep='???',
                 header=True,
                 index=False,
                 index_label=None,
                 encoding='UTF-8',
                 compression=None,
                 decimal='.')

dfX_azure.to_csv('dfx_azure.csv',
                 sep=',',
                 na_rep='???',
                 header=True,
                 index=False,
                 index_label=None,
                 encoding='UTF-8',
                 compression=None,
                 decimal='.')

##############################################################################
##

# Define regression function

def reg_func(features, target, explanation, trade_type, algo):
    feature_count = features.shape[1]
    sample_count = features.shape[0]
    min_samples = 100

    if algo == 'linear':
        regr = linear_model.LinearRegression()
    elif algo == 'ridge':
        regr = linear_model.Ridge(alpha=1.0, random_state = 7)
    elif algo == 'lasso':
        regr = linear_model.Lasso(alpha=0.1, random_state = 7)
    elif algo == 'randomforest':
        regr = RandomForestRegressor(max_depth = feature_count-1, random_state = 7)
    elif algo == 'decisiontree':
        regr = DecisionTreeRegressor(max_features = feature_count-1, random_state=7)
```

```
    elif algo == 'gradientboosting':
        regr = GradientBoostingRegressor(random_state=7)

    X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.30,
random_state=7)

    if sample_count < min_samples:
        to_return = {
                'coefficients': [0] * feature_count,
                'mse_test': 0,
                'r2_test': 0,
                'mse_train': 0,
                'r2_train': 0,
                'algorithm':algo,
                'explanation': explanation,
                'trade_type': trade_type,
                'target':str(target.name),
                'sample_size':sample_count
            }

        print('Not enough samples.')

        return to_return

    regr.fit(X_train, y_train)
    ytrain_pred = regr.predict(X_train)
    y_pred = regr.predict(X_test)

    to_return = {
                'coefficients': regr.coef_ if hasattr(regr, 'coef_') else [0] * feature_count,
                'mse_test': mean_squared_error(y_test, y_pred),
                'r2_test': r2_score(y_test, y_pred),
                'mse_train': mean_squared_error(y_train, ytrain_pred),
                'r2_train': r2_score(y_train, ytrain_pred),
                'algorithm':algo,
                'explanation': explanation,
                'trade_type': trade_type,
                'target':str(target.name),
                'sample_size':sample_count
            }
    """
    print('############################################################')
    print('Explanation: ' + str(explanation))
    print('Trade type: ' + str(trade_type))
    print('Target: ' + str(target.name))
    print('Algorithm: ' + str(algo))
```

```
    print('Coefficients: ' + str(to_return['coefficients']))
    print('Mean squared error: ' + str(to_return['mse']))
    print('R square: ' + str(to_return['r2']))
    print('############################################################')
    """

    print('Finished regression for following:'+ str(algo) + ','
                                      + str(explanation) + ','
                                      + str(trade_type))
    return to_return

# Collect results for overall features and targets

"""
results_ls_all = []

for ls in model_inputs:
    for target in ls[1]:
        results_ls_all.append(reg_func(ls[0],ls[1][target],'Overall Regression',ls[2],'linear'))
"""

# Define regression function for categorical feature breakdowns

def reg_breakdown(input_features,input_target,col,trade_type,algo):
    features = input_features.copy()
    target = input_target.copy()
    breakdowns = features[col].unique().tolist()
    # print(breakdowns)

    reg_results = []
    for breakdown in breakdowns:
        new_features = features[features[col] == breakdown]
        new_target = target[new_features.index.values]
        exp = str(col) + "_" + str(breakdown)
        reg_results.append(reg_func(new_features, new_target, exp, trade_type, algo))
        del new_features, new_target
    return reg_results

# Collect results for each categorical feature

results_ls_breakdown = []
breakdown_cols = ['COUNTRY','BEC','ISIC3']
algorithms = ['linear','lasso','ridge','randomforest','decisiontree','gradientboosting']

for ls in model_inputs:
    for target in ls[1]:
```

```
      for breakdown in breakdown_cols:
         for algorithm in algorithms:
               results_ls_breakdown.append(reg_breakdown(ls[0],
                                          ls[1][target],
                                          breakdown,
                                          ls[2],
                                          algorithm))

result_columns = ['Trade_Type', 'Explanation', 'Target', 'Algorithm',
            'MSE_Test', 'R2_Test', 'MSE_Train', 'R2_Train', 'Sample_Size',
            'Coef_YEAR','Coef_MONTH','Coef_REXR',
            'Coef_REXR_PCT_1','Coef_REXR_PCT_3','Coef_REXR_PCT_6',
            'Coef_CLOSE_STD','Coef_CLOSE-OPEN_STD', 'Coef_HIGH-LOW_STD',
            'Coef_REXR_STD', 'Coef_CLOSE_PCT_1', 'Coef_CLOSE_PCT_3',
'Coef_CLOSE_PCT_6',
            'Coef_ISIC3','Coef_BEC', 'Coef_COUNTRY'
            ]

results_df = pd.DataFrame(index=None, columns = result_columns)

for lol_item in results_ls_breakdown:
   for lod_item in lol_item:
      results_df = results_df.append( {'Trade_Type':lod_item['trade_type'],
                              'Explanation':lod_item['explanation'],
                              'Target':lod_item['target'],
                              'Algorithm':lod_item['algorithm'],
                              'MSE_Test':lod_item['mse_test'],
                              'R2_Test':lod_item['r2_test'],
                              'MSE_Train':lod_item['mse_train'],
                              'R2_Train':lod_item['r2_train'],
                              'Sample_Size':lod_item['sample_size'],

                              'Coef_YEAR':lod_item['coefficients'][0],
                              'Coef_MONTH':lod_item['coefficients'][1],
                              'Coef_REXR':lod_item['coefficients'][2],
                              'Coef_REXR_PCT_1':lod_item['coefficients'][3],
                              'Coef_REXR_PCT_3':lod_item['coefficients'][4],
                              'Coef_REXR_PCT_6':lod_item['coefficients'][5],
                              'Coef_CLOSE_STD':lod_item['coefficients'][6],
                              'Coef_CLOSE-OPEN_STD':lod_item['coefficients'][7],
                              'Coef_HIGH-LOW_STD':lod_item['coefficients'][8],
                              'Coef_REXR_STD':lod_item['coefficients'][9],
                              'Coef_CLOSE_PCT_1':lod_item['coefficients'][10],
                              'Coef_CLOSE_PCT_3':lod_item['coefficients'][11],
                              'Coef_CLOSE_PCT_6':lod_item['coefficients'][12],
                              'Coef_ISIC3':lod_item['coefficients'][13],
```

```
                                'Coef_BEC':lod_item['coefficients'][14],
                                'Coef_COUNTRY':lod_item['coefficients'][15]

                                }, ignore_index=True)

# Drop 'Gizli Veri' from results
results_df = results_df.drop(results_df[(results_df['Trade_Type'] == 'import') &
                                (results_df['Explanation'] == 'ISIC3_33')].index
                        )

results_df = results_df.drop(results_df[(results_df['Trade_Type'] == 'import') &
                                (results_df['Explanation'] == 'BEC_17')].index
                        )

# Replace Country, BEC, ISIC3 Codes with actual names

path = os.path.expanduser("~/Documents/capstone/outputs")
os.chdir(path)

results_df[['Breakdown_Type', 'Breakdown_Code']] = results_df['Explanation'].str.split('_',
n=1, expand=True)
results_mappings = pd.read_csv('result_mappings.csv', sep=',')
results_mappings['Breakdown_Code'] = results_mappings['Breakdown_Code'].astype(int)
results_df['Breakdown_Code'] = results_df['Breakdown_Code'].astype(int)

results_df = pd.merge(results_df, results_mappings, on=['Trade_Type', 'Breakdown_Type',
'Breakdown_Code'], how='left')

result_cols = ['Trade_Type', 'Breakdown_Type', 'Breakdown_Code', 'Breakdown_Name',
'Explanation',
        'Target', 'Algorithm', 'MSE_Test', 'R2_Test', 'MSE_Train', 'R2_Train', 'Sample_Size',
        'Coef_YEAR', 'Coef_MONTH', 'Coef_REXR', 'Coef_REXR_PCT_1',
        'Coef_REXR_PCT_3', 'Coef_REXR_PCT_6', 'Coef_CLOSE_STD',
        'Coef_CLOSE-OPEN_STD', 'Coef_HIGH-LOW_STD',
        'Coef_REXR_STD', 'Coef_CLOSE_PCT_1', 'Coef_CLOSE_PCT_3',
'Coef_CLOSE_PCT_6',
        'Coef_ISIC3', 'Coef_BEC', 'Coef_COUNTRY',
        ]

results_df = results_df[result_cols]

# Save Results to CSV file

results_df.to_csv('reg_results.csv',
                sep=',',
                na_rep='???',
```

```
            header=True,
            index=False,
            index_label=None,
            encoding='UTF-8',
            compression=None,
            decimal='.')

# Calculate run time
end_time = datetime.now()
print('Congrats! Your script has finished executing. Run time:')
print(end_time - start_time)
```