MEF UNIVERSITY

PREDICTING THE RESPONSIBLE DEPARTMENTS FOR THE HUMAN RESOURCES RELATED QUESTIONS BY USING THE TEXT CLASSIFICATION ALGORITHMS

Capstone Project

Yavuz Sancı

İSTANBUL, 2018

MEF UNIVERSITY

PREDICTING THE RESPONSIBLE DEPARTMENTS FOR THE HUMAN RESOURCES RELATED QUESTIONS BY USING THE TEXT CLASSIFICATION ALGORITHMS

Capstone Project

Yavuz Sancı

Advisor: Prof. Dr. Özgür Özlük

İSTANBUL, 2018

MEF UNIVERSITY

Name of the project: Predicting the Responsible Departments for the Human Resources Related Questions by Using the Text Classification Algorithms
Name/Last Name of the Student: Yavuz Sancı
Date of Thesis Defense: 10/09/2018

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

10/09/2018
Prof. Dr. Özgür Özlük

I hereby state that I have examined this graduation project by Yavuz Sancı 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.

10/09/2018

Director
of
Big Data Analytics Program

We hereby state that we have held the graduation examination of Yavuz Sancı and agree that the student has satisfied all requirements.

THE EXAMINATION COMMITTEE

Committee Member	Signature
1. Prof. Dr. 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.

Yavuz Sancı	10/09/2018	Signature

EXECUTIVE SUMMARY

PREDICTING THE RESPONSIBLE DEPARTMENTS FOR THE HUMAN RESOURCES RELATED QUESTIONS BY USING THE TEXT CLASSIFICATION ALGORITHMS

Yavuz Sancı

Advisor: Prof. Dr. Özgür Özlük

SEPTEMBER, 2018, 26 pages

The employees of Yapı Kredi Bank use a help desk system to ask their Human Resources related questions to the employees of the Human Resources departments. The questions are assigned automatically to the relevant departments by the system according to the subjects of the questions. In some cases, the mismatches between the contents and the subjects of the questions may cause the wrong Human Resources department assignments of the questions. Even though the application allows Human Resources employees to redirect the questions to the appropriate Human Resources departments, which are responsible for answering, the response time of these questions lasts longer. This project aims to analyze the content of the Human Resources related questions by using the text classification algorithms to predict the responsible Human Resources departments. Thus, it is aimed to respond to the questions in a much shorter time.

Key Words: Text Classification, Human Resources, Predicting

ÖZET

METİN SINIFLANDIRMA TEKNİKLERİ KULLANARAK İNSAN KAYNAKLARI İLE İLGİLİ SORULAR İÇİN SORUMLU EKİPLERİN TAHMİNLENMESİ

Yavuz Sancı

Tez Danışmanı: Prof. Dr. Özgür Özlük

EYLÜL, 2018, 26 sayfa

Yapı ve Kredi Bankası çalışanları İnsan Kaynakları ile ilgili sorularını bir talep yönetimi sistemi kullanarak İnsan Kaynakları çalışanlarına iletmektedir. Soruların hangi İnsan Kaynakları ekibine sistem tarafından yönlendirileceği; çalışanın sorusunu sorarken seçeceği konu başlığına göre belirlenmektedir. Bazı durumlarda; seçilen konu başlığıyla sorunun içeriği birbiriyle örtüşmediği için uygulama bir takım soruları yanlış İnsan Kaynakları ekiplerine yönlendirmektedir. Her ne kadar; İnsan Kaynakları çalışanları kendilerinin onayına düşen bu soruları yanıtlamakla sorumlu olan diğer İnsan Kaynakları ekiplerine yönlendirebiliyor olsalar da; bu durum soruların çözüm sürelerinin uzamasına sebep olmaktadır. Bu çalışma; metin sınıflandırma teknikleri kullanarak İnsan Kaynakları ile ilgili soruların metin içeriklerinin analiz edilmesini ve sorulara cevap vermekle sorumlu İnsan Kaynakları departmanlarının tahminlenmesini kapsamaktadır. Bu sayede, banka çalışanlarının İnsan Kaynakları'na ilettiği sorulara çok daha kısa süre içerisinde yanıt verilmesi hedeflenmektedir.

Anahtar Kelimeler: Metin Sınıflandırma, İnsan Kaynakları, Tahminleme

TABLE OF CONTENTS

.

1. INTRODUCTION

The Human Resources help desk applications are essential for delivering the HR services of an organization efficiently to its employees in a digital environment. The implementation of a centralized help desk application across the organization provides handling the questions, requests with transparency and helps HR to classify the category of these questions and requests. The service level agreement (SLA) metrics, which can be gathered by the help desk applications, can also measure the performance of each HR department according to the responsiveness and the response time. The responsiveness of the help desk applications can be optimized by establishing an automated triage system which sorts the incoming requests by priority. [1] The performance rating of the departments can be reviewed periodically in order to make decisions for the improvement of their performance scores.

The employees of an organization can make any HR related request and receive feedbacks from HR easily by using the self-service screens of a centralized HR help desk application. If the employees get their answers on time with the sufficient and clear explanations which are consistent with the culture and the business objectives of the company, the employees can stay more organized. Moreover, if the HR services, which are delivered through the help desk application channel, are appreciated by the majority of the employees; this will also increase the employee engagement which is a priority for most of the companies.

In Yapı Kredi, an HR help desk application is available for the employees of the entire company. When an employee wants to make a request by using the application; first an appropriate subject should be selected from a list of subjects and then a text field, which contains information about the question, should be filled in by the employee. After the creation of the record, the application assigns it to the relevant HR department according to the subject which is previously selected by the employee.

In some cases, the employees may select the wrong subjects for their questions or requests. These will cause inappropriate assignments. Despite the fact that the application allows HR employees to transfer the questions to the appropriate HR departments, the response time of these records lasts longer.

The aim of this project is to work on predicting the responsible HR departments according to the content of the questions by using text classification algorithms in a help desk application.

2. ABOUT THE DATA

The dataset of the project is retrieved from the database which stores the data of Human Resources help desk application of Yapı Kredi Bank. It consists of the HR related questions and requests which were created by the employees of the bank in the first half of 2018. The personal information of the employees and any HR sensitive information were eliminated during the preparation of the dataset. The dataset has 6389 rows and 5 columns in total. The columns of the dataset and their descriptions are stated below:

• RequestId	: It contains a unique value for each record of the dataset.
• Department	: The department responsible for answering the question
• Subject	: The subject of the question selected by the employee
• Description	: The content of the question defined by the employee
• RequestDate	: The creation date of the question

A sample row, which is extracted from the dataset, is stated below. (Figure 1) In this example, an employee requests the salary slips of the last three months in order to deliver them to the Netherlands Consulate. Since the subject of the request is selected as "Bordro Talepleri" by the employee; the request is automatically assigned to the department of "Kurumsal İK Yönetimi/Bordro İşlemleri".

Figure 1: A Sample Row Which is Extracted from the Dataset

RequestId	Department	Subject	Description	RequestDate
12584	Kurumsal İK Yönetimi/Bordro İşlemleri	Bordro Talepleri	Merhaba, Hollanda Konsolosluğu'na sunulmak üzere son 3 ay bordrolarımın tarafıma gönderilmesini rica ederim.	11.04.2018 09:21:44

In the analysis of the dataset, Department column is selected as the target variable and Description column, which contains the detailed explanations of the questions, is taken into consideration for extracting the new features of tokens.

2.1. About Yapı Kredi

Yapı Kredi, which is the fourth largest bank in Turkey as of 2017 due to the number of assets, is one of 10 most valuable brands in Turkey. The customer-centric core banking and innovative banking technologies are the main focus of Yapı Kredi for achieving profitable and sustainable growth together with increasing customer satisfaction. The Bank's main shareholder is Koç Financial Services (KFS) with 81.8% ownership. KFS is a 50%-50% joint venture between Koç Group and UniCredit Group. The remaining 18.2% is publicly traded on Borsa Istanbul.

Yapı Kredi has 866 branches and 18839 employees in total. All these employees have access to HR help desk application for making their HR related requests by using the intranet of the bank. Therefore, the HR help desk system, which is used in Yapı Kredi, is an important interaction channel between the employees and HR departments.

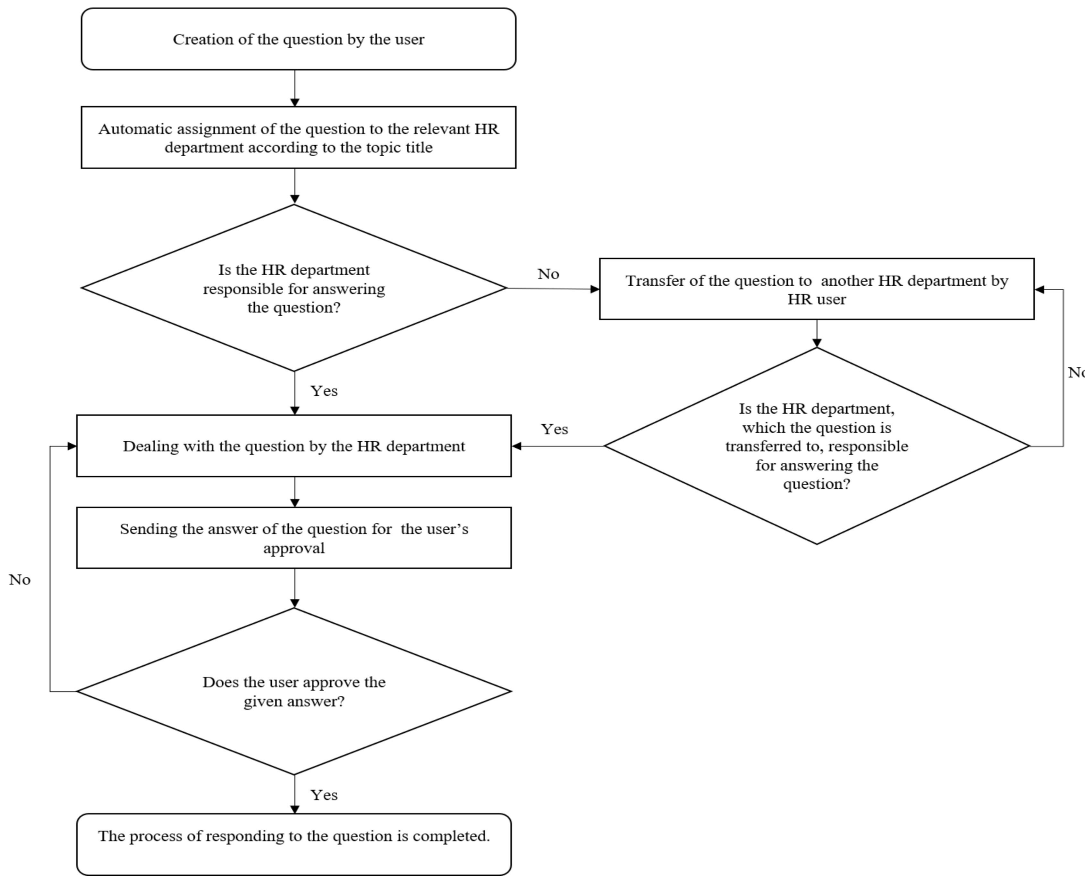
3. PROJECT DEFINITION

3.1. Problem Statement

The assignment of a question to an HR department which is not responsible for answering is a severe problem which increases the response time. The assignments of the questions are made automatically by the HR help desk application according to the subject which the employees choose from a list box during the creation of the question records. A wrong assignment occurs when there is a mismatch between the selected subject and the content of the question. Approximately 15% of all questions are assigned to the HR departments which are not responsible for answering due to the employees' wrong choices of the subjects. The HR employees, who encounter these problems, should transfer the questions to the departments which should be responsible for responding to them. In some cases, there may still be an incorrect assignment because the assignment is determined according to one's own initiative.

As shown in the following flowchart (Figure 2); the process of responding to a question lasts longer when there is a wrong department assignment of the related question. For a question which is assigned to an HR department; the related HR department is expected to reply to the question within 3 working days according to the SLA policies which are implemented in Yapı Kredi. If the question is not related to the HR department; the HR department should transfer the question to the appropriate department within 1 working day. If the department, to which the question is transferred, is responsible for answering the question, the maximum response time is extended from 3 working days to 4 working days.

Figure 2: The Flow Chart of HR Help Desk Application in Yapı Kredi


Creation of the question by the user
Automatic assignment of the question to the relevant HR department according to the topic title
Is the HR department responsible for answering the question?
No
Transfer of the question to another HR department by HR user
No
Yes
Yes
Dealing with the question by the HR department
Is the HR department, which the question is transferred to, responsible for answering the question?
Sending the answer of the question for the user's approval
No
Does the user approve the given answer?
Yes
The process of responding to the question is completed.

3.2. Project Objectives

The main objective of the project is to develop a precise algorithm that labels the responsible HR departments according to the content of the questions. Measuring the performance of the different algorithms, which are used for classifying the questions, is another objective of this project.

3.2. Project Scope

The scope of the project covers constructing a precise model which classifies the questions according to their relevant HR departments. First, the feature extraction methods are applied to the text contents of the questions to form the bag-of-words presentations. After the process of feature extraction, this study covers the implementation of the different classification algorithms to the final version of the dataset and compares the performance of each classification algorithm.

4. LITERATURE REVIEW

4.1. Text Classification Methods

Text classification is an essential topic in Natural Processing Language, in which the pre-defined categories should be labeled to the text documents. It is used in many applications such as web search, information retrieval, ranking and document classification.

Today, choosing the best possible machine learning classifiers for text classification is one of the popular research topics. The performance of Neural Network approaches is better than the performance of the traditional Bag-of-Words models for text classifications since Neural Networks focus on the semantic relation between the words and their order in a sentence. [2] Character-Level Convolutional Networks perform well on the large datasets without the need for the words for text classification. Character-Level Convolutional Networks can also identify the misspellings, which are generated by the users, to conduct better text classifications than the traditional models such as the Bag of Words models with their TF-IDF weights. [3]

On the other hand, FastText; which is a non-deep learning model, performs faster than the Character-Level Convolutional Networks and has the same accuracy scores as Character-Level Convolutional Networks. FastText maps each word into a bag of n-grams features with an additional hashing algorithm and averages the features of the all words to get a good representation of sentences. [4]

Hierarchical neural network, which is another method for text classification, uses the word-level and the sentence-level attention mechanisms to weight the importance of the words in the sentences and the importance of the sentences in the documents respectively. [5]

Paragraph vectors can predict the next word in a context by benefiting from the semantics which is captured by themselves. Since the word order and the semantics of words in a paragraph are taken into consideration, paragraph vector can be an alternative method for text classification. [6]

4.2. The Impact of Machine Learning in Human Resources

In Human Resources, several machine learning algorithms have been used to analyze the employee turnover such as Naive Bayes, Random Forest, and Support Vector Machine. [7] Today, employee turnover is a significant problem which the companies should deal with. It is a measurement of how long the employees work for a company and how often the company has to replace them. The companies should hire and train new employees instead of the ones who leave the job so they can fill the vacancy positions. Leaving of the employees can also change the organizational structure of a company because there may be changes in the job responsibilities for some employees. If a company has a high employee turnover rate, much workforce loss and cost occur. Even worse, the reputation of the company can be negatively affected. Therefore, it is crucial for the companies to discover the most important reasons affecting the employee turnover. It can be achieved by predicting the employees, who are more likely to leave the company.

The model of Intelligent Human Resources Systems predicts the employees who might leave in the future by using the Support Vector Machine algorithms. [8] The Random Forest model, which was constructed by Dilip Singh Sisodia et. Al, has a good performance in predicting the employees who may leave the company. This model can also give insights about the most critical reasons affecting the employee turnover. According to the model; the employees, who cannot get promotions for a long time, and the employees, who work for longer periods of time, are more likely to leave. [9]

5. METHODOLOGY

The first step of this study is to analyze the distribution of the questions according to their related Human Resources Department in order to check the existence of imbalanced class distribution. In the case of the presence of the imbalanced data; the resampling method, Smote, is planned to be applied to the dataset to avoid overfitting by equalizing the number of the minority classes to the number of the majority class. The second step is to extract new numerical feature vectors by transforming the text content of the questions into the bag-of-words models. Each word, which resides in a question, is presented by a feature and the importance of each word is calculated by using the term frequency-inverse document frequency (TF-IDF) weighting. In the third step, the different machine learning algorithms are executed to predict the relevant HR department of the questions. Naive Bayes, Stochastic Gradient Descent and Random Forest classification algorithms are constructed, and the total accuracy score of each algorithm is compared with each other to find the one which has the best accuracy score. For each algorithm; the test accuracy and the train accuracy scores of the related algorithm are also compared to check whether an overfitting exists. The K-Fold cross-validation techniques are also used to evaluate the performance of each algorithm by reducing the variance and bias of the models. The parameters of the algorithm, which performs better than the other ones according to the test accuracy score, are tuned by using the grid search method to improve the accuracy. Finally, a confusion matrix of the best algorithm is constructed to see the discrepancies between predicted and actual labels. All these steps are executed on the JetBrains PyCharm Community Edition platform by using the Python programming language.

5.1. Using Bag of Words with TF-IDF weights

In machine learning algorithms, the text data should be converted into a bag-words model which encodes its meaning as much as possible. In this study, the Bag of Words model is constructed with TF-IDF weights.

TF-IDF is the abbreviation of term frequency-inverse document frequency. It is a statistical measure to evaluate how important a word is to a document in a collection. The term frequency refers to the number of times in which a word occurs in a document. The inverse document frequency is a measure of how much information is provided by the word according to its frequency within all the documents. The inverse document frequency is calculated by taking the logarithm of the inverse fraction of the documents that contain the word, which is the ratio of the total number of documents to the number of documents containing the term. TF-IDF weight of a word is equal to the product of the term frequency and the inverse document frequency of the related word. [10]

According to the TF-IDF weighting; as the number of times, which a word appears in a question, increases; the importance of the word increases. On the other hand; the importance of the word decreases as the number of times, which the word appears in all questions, increases. For example; the following two sentences can be considered:

Sentence 1: “Are you doing what he is doing?”

Sentence 2: “He is doing a good job.”

After removing the stop words, which are often functional words that contribute to the meaning of the sentence through grammar; the remaining each word is weighted according to TF-IDF as shown below. (Figure 3)

Figure 3: Distribution of Words According to TF-IDF Weighting

	doing	**good**	**job**
Sentence 1	1	0	0
Sentence 2	0,44943642	0,6316672	0,6316672

As the TF-IDF weights of the words, which reside in two sentences, are compared; we can see that the word “doing”, which occurs in both of the sentences, is penalized in the second sentence because it appears twice in the first sentence.

5.2. Multinomial Naive Bayes

Multinomial Naive Bayes is a customized version of the Naive Bayes algorithm which can be used for classification with discrete features such as the word counts of the documents. In text classification; a simple Naive Bayes algorithm only classifies a document according to the presence or the absence of the specific words and the frequency information of the words isn’t considered. On the other hand, the frequency information of the words, which occurs in a document, is considered in Multinomial Naive Bayes to make better classifications. [11] Multinomial Naive Bayes algorithm assumes that each feature is independent of each other given the class. The separate learning of the parameters of each feature eases the training process especially when the dataset, which is analyzed, has a large number of the attributes. Most of the real-world datasets, which the text classification methods are implemented, have more accurate results if they have vocabularies of huge numbers of the words. Therefore, Multinomial Naive Bayes is widely used in many studies which involve text classification. The disadvantages of using this algorithm are that the semantic relation between the words in a document are ignored and poor weights are selected for the decision boundary if a particular class has more records than the other classes in a dataset.

5.3. Stochastic Gradient Descent

The objective of the gradient descent algorithm is to minimize a cost function which is calculated by summing up the cost function of each sample of the training set. The gradient descent becomes a computationally very expensive procedure in the case of the existence of a huge training set. A modified version of the gradient descent algorithm, which is called Stochastic Gradient Descent, is more appropriate to be implemented for much bigger training sets.

In a Basic Gradient Descent algorithm; the cost function is equal to the sum of the one half of the average square error of the hypothesis on the m training examples. As we run different iterations of the gradient descent from an initialization point by updating the values of the parameters, the parameters will be directed to a global minimum. If the value of m, which is the total number of training examples, is too large, the computation of the gradients will be very expensive because the total number of training examples will be used to calculate the gradient in a single iteration. The total number of the training examples, which are used to calculate the gradient, is also called as “batch”.

In contrast to basic Gradient Descent algorithm; Stochastic Gradient Descent doesn’t need to look at the total number of examples in each iteration to calculate the gradient. [12] In order to reach to the global minimum; a single training example, which is randomly selected, will be enough for calculating the gradient on each iteration. The algorithm will scan through all the training examples and on each iteration; a gradient descent step is taken by looking at each example with respect to its cost. When the scanning through all training examples is completed, it may be necessary to repeat the entire scanning operation for n times depending on the size of the training set.

Since the Stochastic Gradient Descent algorithm has an efficient performance of scaling up extensive datasets, it is successfully applied for text classification methods which analyze vast amounts of text contents. Dealing with the several hyperparameters and the sensitiveness to feature scaling can be considered as the main disadvantages of the Stochastic Gradient Descent.

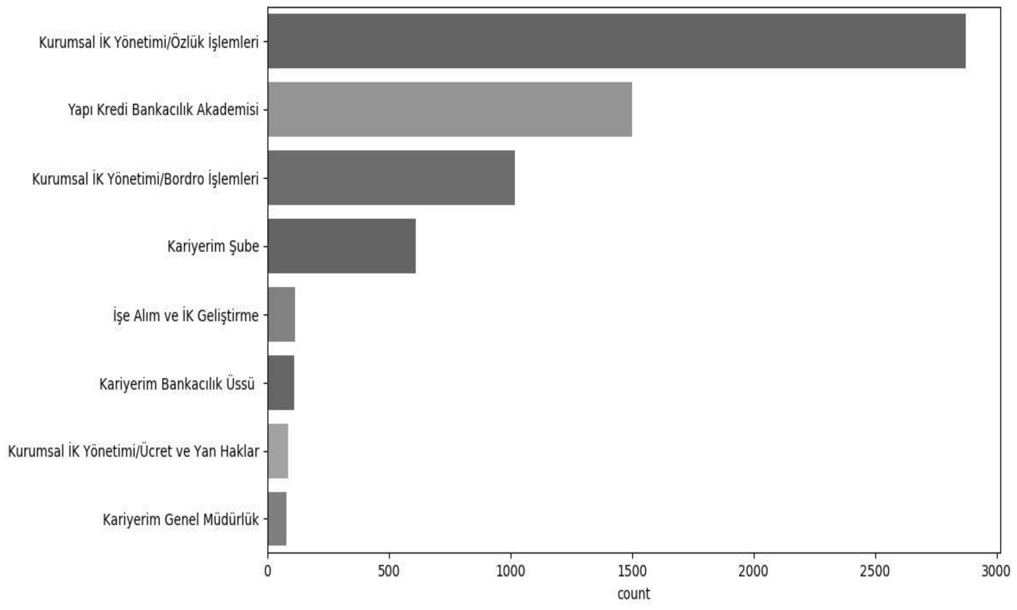
5.4. Random Forest

Random Forest is a supervised learning algorithm which can be used both for regression and classification problems. The ease of use and the flexibility of the algorithm has made Random Forest as one of the most widely used Machine Learning techniques. As compared to the Stochastic Gradient Descent algorithm, which is one of the Machine Learning techniques which are used in this study; hyperparameter tuning is not necessary for Random Forest. It is an ensemble of the decision trees which are combined to get more accurate results. Unlike the decision trees; Random Forest searches for the best feature from a randomly selected subset of features instead of choosing the most important feature in the node splits. This randomness provides diversity which produces better results than the decision trees. In Random Forest models, the relative importance of each feature can be analyzed after the prediction of the classes. The insignificant features can be removed by observing the relative importance values of the features to avoid overfitting. The main disadvantage of the Random Forest algorithm is that a large number of trees makes the algorithm to perform slowly despite the fact that more decision trees allow Random Forest to make more accurate predictions. [13]

6. RESULTS

The distribution of the questions is displayed according to their related HR departments in the following chart. It is evident that there is an imbalanced distribution of the classes in the dataset. According to the chart; Kurumsal İK Yönetimi/Özlük İşlemleri is responsible for most of the questions (45%) followed by Yapı Kredi Bankacılık Akademisi (23,5%) and Kurumsal İK Yönetimi/Bordro İşlemleri (16%). The other departments are only responsible for 15,5 % of the total questions.

Figure 4: The Distribution of the Questions According to HR Departments



The train and test accuracy scores of the Machine Learning algorithms, which are applied to the original dataset before applying Smote function, are stated below. (Figure 4) According to these accuracy results; the differences between the initial test accuracy scores and the initial train accuracy scores are large for all the algorithms except Multinomial Naive Bayes before applying K-Fold cross-validation techniques. The probability of an overfitting problem is very high, especially for the Random Forest. The main reason for overfitting can be the imbalanced classes. 10 folds cross-validations are applied to each algorithm to overcome the overfitting problem. Although these cross-validation techniques decrease the accuracy scores of each algorithm, the overfitting problem is avoided by the decrease in the difference between the train accuracy score and the test accuracy score for each algorithm.

Figure 5: The Accuracy Scores of the M.L. Algorithms Before Applying Smote

Algorithm	Train Accuracy Score	Test Accuracy Score	Train Accuracy Score with 10 Folds Cross Validations	Test Accuracy Score with 10 Folds Cross Validations
Multinomial Naive Bayes	0,7455	0,7147	0,6903	0,675
Stochastic Gradient Descent	0,9088	0,8378	0,8238	0,8237
Random Forest	0,9917	0,7887	0,7907	0,7512
Stochastic Gradient Descent with Grid Search	0,9917	0,8753	0,8647	0,8466

In order to construct an equal distribution of the classes; the resampling method, Smote, is applied to the dataset. This method equalizes the number of each minority class to the number of the majority class by generating the new instances of the minority classes. The majority class, which is presented by Kurumsal İK Yönetimi/Özlük İşlemleri, has 2872 records. After the oversampling process, the total number of records increases from 6389 to 22996 because the number of the records of each of the other 7 minority classes also increases to 2872. The train and test accuracy scores of the Machine Learning algorithms, which are applied to the dataset after the process of oversampling, are stated below. (Figure 4)

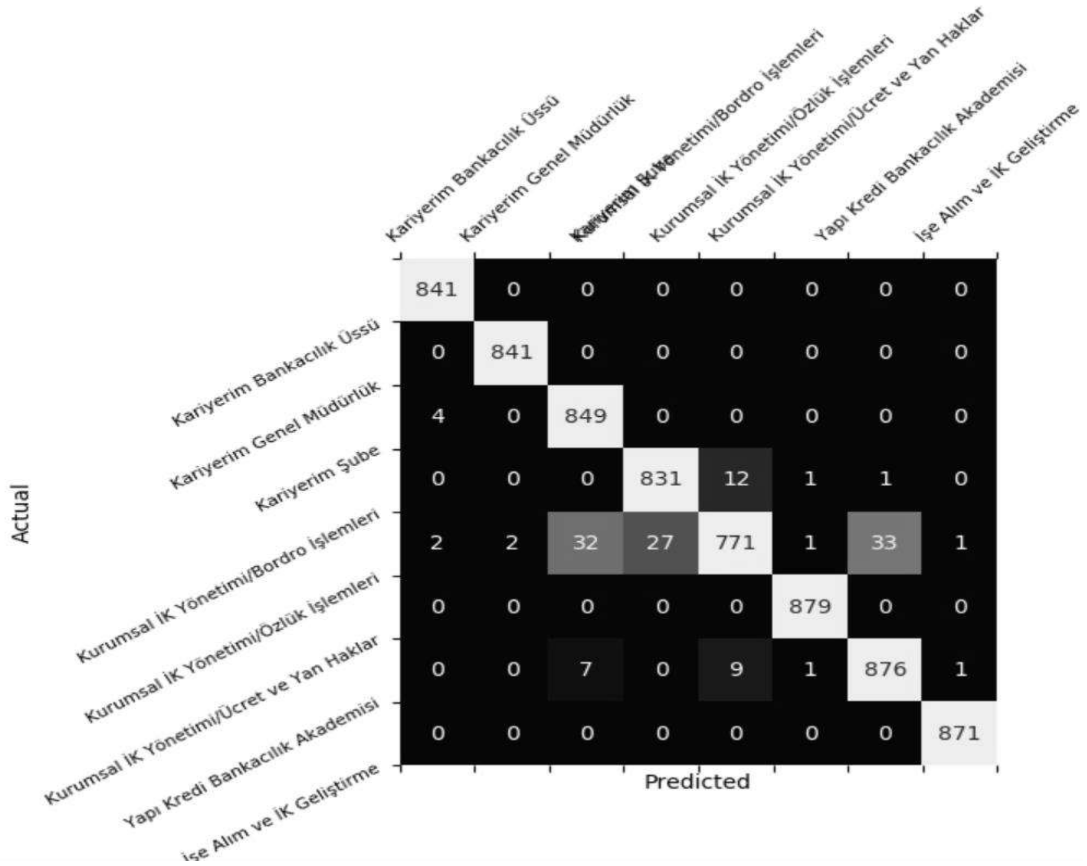
Figure 6: The Accuracy Scores of the M.L. Algorithms After Applying Smote

Algorithm	Train Accuracy Score	Test Accuracy Score	Train Accuracy Score with 10 Folds Cross Validations	Test Accuracy Score with 10 Folds Cross Validations
Multinomial Naive Bayes	0,9734	0,953	0,9538	0,9272
Stochastic Gradient Descent	0,9624	0,9433	0,9478	0,9366
Random Forest	0,9987	0,9502	0,9553	0,9219
Multinomial Naive Bayes with Grid Search	0,9949	0,9806	0,9787	0,961

After the resampling process, the performance of all algorithms increased significantly. All of the constructed models have very high accuracy scores and the difference between the train accuracy score, and the test accuracy score is acceptable for each of the algorithms.

The confusion matrix of the Multinomial Naive Bayes with Grid Search model, which has the best test accuracy score, is displayed below. (Figure 5) The true positive rate (recall) of all of the classes have high values. Only, "Kurumsal İK Yönetimi/Özlük İşlemleri" class has a slightly lower recall than the recall of the other classes. The false predictions of this class are mostly the actual members of Kariyerim Şube, Kurumsal İK Yönetimi/Bordro İşlemleri, and Yapı Kredi Bankacılık Akademisi.

Figure 7: Confusion Matrix of Multinomial Naive Bayes with Grid Search



	Kariyerim Bankacılık Üssü	Kariyerim Genel Müdürlük	Kariyerim Şube	Kurumsal İK Yönetimi/Bordro İşlemleri	Kurumsal İK Yönetimi/Özlük İşlemleri	Kurumsal İK Yönetimi/Ücret ve Yan Haklar	Yapı Kredi Bankacılık Akademisi	İşe Alım ve İK Geliştirme
Kariyerim Bankacılık Üssü	841	0	0	0	0	0	0	0
Kariyerim Genel Müdürlük	0	841	0	0	0	0	0	0
Kariyerim Şube	4	0	849	0	0	0	0	0
Kurumsal İK Yönetimi/Bordro İşlemleri	0	0	0	831	12	1	1	0
Kurumsal İK Yönetimi/Özlük İşlemleri	2	2	32	27	771	1	33	1
Kurumsal İK Yönetimi/Ücret ve Yan Haklar	0	0	0	0	0	879	0	0
Yapı Kredi Bankacılık Akademisi	0	0	7	0	9	1	876	1
İşe Alım ve İK Geliştirme	0	0	0	0	0	0	0	871

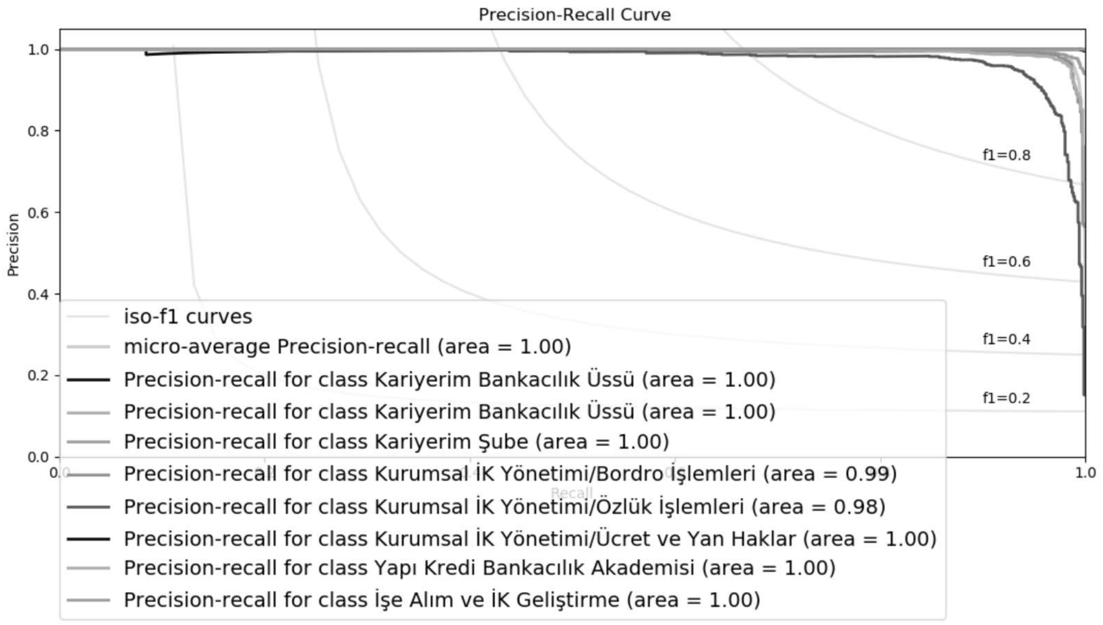
The classification report, which shows the values of the primary classification metrics of the Multinomial Naive Bayes model with Grid Search, is listed below. (Figure 8) Since we get an equal distribution of classes by applying the oversampling process, the support value of each class is close to each other. The high values of precision, recall, and F1-score are the main indicators of the excellent performance of the model.

Figure 8: The Classification Report for Naive Bayes with Grid Search

Class	Precision	Recall	F1-Score	Support
Kariyerim Bankacılık Üssü	0,99	1,00	1,00	841
Kariyerim Genel Müdürlük	1,00	1,00	1,00	841
Kariyerim Şube	0,96	1,00	0,98	853
Kurumsal İK Yönetimi/Bordro İşlemleri	0,97	0,98	0,98	845
Kurumsal İK Yönetimi/Özlük İşlemleri	0,97	0,89	0,93	869
Kurumsal İK Yönetimi/Ücret ve Yan Haklar	1,00	1,00	1,00	879
Yapı Kredi Bankacılık Akademisi	0,96	0,98	0,97	894
İşe Alım ve İK Geliştirme	1,00	1,00	1,00	871
avg / total	0,98	0,98	0,98	6893

The precision-recall curve for each class is represented in the diagram which is displayed below. (Figure 9) Having a large area under the curve represents both high recall and high precision. Since all of the classes have large areas under their related precision-recall curves in the Naive Bayes model with Grid Search, it can be concluded that the model performs well in the prediction of the classes.

Figure 9: Precision-Recall Curves of Naive Bayes with Grid Search



7. DELIVERED VALUE AND FURTHER STEPS

In this study, the accurate predictions of the Human Resources departments, which are responsible for answering the Human Resources related questions through a channel of Human Resources help desk application, were achieved by using the text classification methods. The resampling process made all of the three different machine learning algorithms of this study to have very high overall accuracy scores. In the next step, the accurate model of this study can be integrated to the existing help desk application in order to assign the questions to the related responsible Human Resources department automatically by analyzing the content of the questions. The possible integration of the model will also eliminate the need for selecting a subject during the creation phase of the question in the help desk application.

The Human Resources help desk application, which is used in Yapı Kredi, was developed on Microsoft’s .NET Framework by using the C# programming language. Microsoft's .NET Framework can invoke Python methods within a C# code by using IronPython, which is an open-source implementation of Python programming language on .NET framework. IronPython uses the advantage of .NET’s dynamic language runtime (DLR) feature which provides a set of services to the static programming languages for supporting dynamic programming languages. Many programming functionalities are executed at runtime for dynamic programming languages such as Python and R. On the other hand, static programming languages such as C# and Java perform these functionalities in the compilation phase. On the .NET Framework, the Python code of this study can be accessed from the C# code of the help desk application by using the necessary libraries of IronPython.

The questions of the dataset of this study were in Turkish. Therefore, the future use of a Turkish vocabulary will be beneficial for stemming, semantic reasoning and specifying the stop words in Turkish to make better text classifications.

REFERENCES

[1] Engelhard, M., et al. (2018). Optimising mHealth helpdesk responsiveness in South Africa: towards automated message triage. BMJ global health, 3(Suppl 2), e000567.

[2] Lai, Siwei, et al. Recurrent Convolutional Neural Networks for Text Classification. AAAI. Vol. 333. 2015.

[3] Zhang, X., et al. (2015). Character-level convolutional networks for text classification. In Advances in neural information processing systems (pp. 649-657).

[4] Joulin, A., et al. (2016). Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.

[5] Yang, Z., et al. (2016). Hierarchical attention networks for document classification. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 1480-1489).

[6] Le, Q., & Mikolov, T. (2014, January). Distributed representations of sentences and documents. In International Conference on Machine Learning (pp. 1188-1196).

[7] Ajit, P. (2016). Prediction of employee turnover in organizations using machine learning algorithms. algorithms, 4(5), C5." algorithms 4.5 (2016): C5.

[8] Cahyani, A. D., & Budiharto, W. (2017, February). Modeling Intelligent Human Resources Systems (IRHS) using Big Data and Support Vector Machine (SVM). In Proceedings of the 9th International Conference on Machine Learning and Computing (pp. 137-140). ACM.

[9] Sisodia, D. S., et al. (2017, November). Evaluation of machine learning models for employee churn prediction. In Inventive Computing and Informatics (ICICI), International Conference on (pp. 1016-1020). IEEE.

[10] Hackeling, G. (2014). Mastering Machine Learning with Scikit-learn. Packt Publishing Ltd.

[11] McCallum, A., & Nigam, K. (1998, July). A comparison of event models for naive bayes text classification. In AAAI-98 workshop on learning for text categorization (Vol. 752, No. 1, pp. 41-48).

[12] Brownlee, J. (2018). Gradient Descent For Machine Learning. Retrieved from https://Machinelearningmastery.com/Gradient-Descent-for-Machine-Learning/

[13] Donges, N. (2018). The Random Forest Algorithm – Towards Data Science. Retrieved from https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd

APPENDIX

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from imblearn.over_sampling import SMOTE
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_predict
from sklearn import metrics

dataset = pd.read_excel('IKya_Sor_Talepler.xls')#read from excel to data
frame
dataset.info() #show summary of dataset
dataset.describe()
dataset.head(10)#return first 10 rows
dataset.isnull().any()#check if there is any null value in dataset
df_x = dataset['DESCRIPTION']
df_mainGroup = dataset['DEPARTMENT']
df_label = pd.DataFrame(df_mainGroup)
codes = pd.Categorical(df_label['DEPARTMENT']).codes
df_label["CATEGORYCODE"] = pd.Categorical(df_label['DEPARTMENT']).codes
cat_columns = df_label.select_dtypes(['category']).columns
df_label[cat_columns] = df_label[cat_columns].apply(lambda x:
x.cat.codes)
df_y = df_label["CATEGORYCODE"]
# TF-IDF
tfidf_vect = TfidfVectorizer()
df_x__tfidf = tfidf_vect.fit_transform(df_x)
# Resampling process to prevent imbalance between classes
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_sample(df_x__tfidf, df_y)
# Plot number of departments before resampling
fig = plt.figure(figsize = (8,6))
descending_order =
df_mainGroup.value_counts().sort_values(ascending=False).index
ax =
sns.countplot(data=pd.DataFrame(df_mainGroup),y="DEPARTMENT",order=descen
ding_order)
plt.show()
#Construction of models without applying SMOTE
#train-test split
x_train, x_test, y_train, y_test = train_test_split(df_x__tfidf, df_y,
test_size = 0.3, random_state = 0)
#Multinominal Naive Bayes
clf_NB = MultinomialNB().fit(x_train, y_train)
predictions_NB_test=clf_NB.predict(x_test)
print("Test Accuracy score of Multinomial Naive Bayes before applying
Smote is " +
str(round(accuracy_score(y_test, predictions_NB_test),4)))
cv_predicted_test = cross_val_predict(clf_NB, x_test, y_test, cv=10)
print("Test Accuracy score of Multinomial Naive Bayes with 10 folds cross
validation before applying Smote is " +
str(round(metrics.accuracy_score(y_test, cv_predicted_test),4)))
predictions_NB_train=clf_NB.predict(x_train)
print("Train Accuracy score of Multinomial Naive Bayes before applying
Smote is " +
str(round(accuracy_score(y_train, predictions_NB_train),4)))
cv_predicted_train = cross_val_predict(clf_NB, x_train, y_train, cv=10)
print("Train Accuracy score of Multinomial Naive Bayes with 10 folds
cross validation before applying Smote is " +
str(round(metrics.accuracy_score(y_train, cv_predicted_train),4)))
#SGD
clf_SGD = SGDClassifier(loss='hinge', penalty='l2', alpha=1e-3,
max_iter=5, random_state=42).fit(x_train, y_train)
predictions_SGD_test=clf_SGD.predict(x_test)
print("Test Accuracy score of Stochastic Gradient Descent before applying
Smote is " +
str(round(accuracy_score(y_test, predictions_SGD_test), 4)))
cv_predicted_test = cross_val_predict(clf_SGD, x_test, y_test, cv=10)
print("Test Accuracy score of Stochastic Gradient Descent with 10 folds
cross validation before applying Smote is " +
str(round(metrics.accuracy_score(y_test, cv_predicted_test),4)))
predictions_SGD_train=clf_SGD.predict(x_train)
print("Train Accuracy score of Stochastic Gradient Descent before
applying Smote is " +
str(round(accuracy_score(y_train, predictions_SGD_train), 4)))
cv_predicted_train = cross_val_predict(clf_SGD, x_train, y_train, cv=10)
print("Train Accuracy score of Stochastic Gradient Descent with 10 folds
cross validation before applying Smote is " +
str(round(metrics.accuracy_score(y_train, cv_predicted_train),4)))
#Random Forest
clf_RF = RandomForestClassifier().fit(x_train, y_train)
predictions_RF_test=clf_RF.predict(x_test)
print("Test Accuracy score of Random Forest before applying Smote is " +
str(round(accuracy_score(y_test, predictions_RF_test), 4)))
cv_predicted_test = cross_val_predict(clf_RF, x_test, y_test, cv=10)
print("Test Accuracy score of Random Forest with 10 folds cross
validation before applying Smote is " +
str(round(metrics.accuracy_score(y_test, cv_predicted_test),4)))
predictions_RF_train=clf_RF.predict(x_train)
print("Train Accuracy score of Random Forest before applying Smote is " +
str(round(accuracy_score(y_train, predictions_RF_train), 4)))
cv_predicted_train = cross_val_predict(clf_RF, x_train, y_train, cv=10)
print("Train Accuracy score of Random Forest with 10 folds cross
validation before applying Smote is " +
str(round(metrics.accuracy_score(y_train, cv_predicted_train),4)))
#Grid Search for improving the performance of Stochastic Gradient Descent
sorted(clf_SGD.get_params().keys()) #displaying the all possible
parameters of SGD
parameters = {
    'loss': ('log', 'hinge'),
    'penalty': ['l1', 'l2', 'elasticnet'],
    'alpha': [0.001, 0.0001, 0.00001, 0.000001]
}
gs_clf = GridSearchCV(clf_SGD, parameters, n_jobs=-1, scoring='accuracy')
gs_clf = gs_clf.fit(x_train, y_train)
predictions_gs=gs_clf.predict(x_test)
```

```
print("The best score of Stochastic Gradient Descent with Grid Search
before applying Smote is " +  str(round(gs_clf.best_score_,4)))
print("Best parameters set found on Stochastic Gradient Descent
algoritm:")
print(gs_clf.best_params_)
print( "Test Accuracy score of Stochastic Gradient Descent with Grid
Search before applying Smote is " +
str(round(accuracy_score(y_test,predictions_gs),4)))
predictions_gs_train=gs_clf.predict(x_train)
print( "Train Accuracy score of Stochastic Gradient Descent with Grid
Search before applying Smote is " +
str(round(accuracy_score(y_train,predictions_gs_train),4)))
cv_predicted_train = cross_val_predict(gs_clf, x_train, y_train, cv=10)
print("Train Accuracy score of Stochastic Gradient Descent with Grid
Search and 10 folds cross validation before applying Smote is " +
str(round(metrics.accuracy_score(y_train, cv_predicted_train),4)))
cv_predicted_test = cross_val_predict(gs_clf, x_test, y_test, cv=10)
print("Test Accuracy score of Stochastic Gradient Descent with Grid
Search and 10 folds cross validation before applying Smote is " +
str(round(metrics.accuracy_score(y_test, cv_predicted_test),4)))
#Construction of models after applying SMOTE
#train-test split
x_train, x_test, y_train, y_test = train_test_split(X_res, y_res,
test_size = 0.3, random_state = 0)
#Multinominal Naive Bayes
clf_NB = MultinomialNB().fit(x_train, y_train)
predictions_NB_test=clf_NB.predict(x_test)
print("Test Accuracy score of Multinomial Naive Bayes after applying
Smote is " +
str(round(accuracy_score(y_test, predictions_NB_test), 4)))
cv_predicted_test = cross_val_predict(clf_NB, x_test, y_test, cv=10)
print("Test Accuracy score of Multinomial Naive Bayes with 10 folds cross
validation after applying Smote is " +
str(round(metrics.accuracy_score(y_test, cv_predicted_test),4)))
predictions_NB_train=clf_NB.predict(x_train)
print("Train Accuracy score of Multinomial Naive Bayes after applying
Smote is " +
str(round(accuracy_score(y_train, predictions_NB_train), 4)))
cv_predicted_train = cross_val_predict(clf_NB, x_train, y_train, cv=10)
print("Train Accuracy score of Multinomial Naive Bayes with 10 folds
cross validation after applying Smote is " +
str(round(metrics.accuracy_score(y_train, cv_predicted_train),4)))
#SGD
clf_SGD = SGDClassifier(loss='hinge', penalty='l2', alpha=1e-3,
max_iter=5, random_state=42).fit(x_train, y_train)
predictions_SGD_test=clf_SGD.predict(x_test)
print("Test Accuracy score of Stochastic Gradient Descent after applying
Smote is " +
str(round(accuracy_score(y_test, predictions_SGD_test), 4)))
cv_predicted_test = cross_val_predict(clf_SGD, x_test, y_test, cv=10)
print("Test Accuracy score of Stochastic Gradient Descent with 10 folds
cross validation after applying Smote is " +
str(round(metrics.accuracy_score(y_test, cv_predicted_test),4)))
predictions_SGD_train=clf_SGD.predict(x_train)
print("Train Accuracy score of Stochastic Gradient Descent after applying
Smote is " +
str(round(accuracy_score(y_train, predictions_SGD_train), 4)))
cv_predicted_train = cross_val_predict(clf_SGD, x_train, y_train, cv=10)
print("Train Accuracy score of Multinomial Stochastic Gradient Descent
with 10 folds cross validation after applying Smote is " +
str(round(metrics.accuracy_score(y_train, cv_predicted_train),4)))
#Random Forest
clf_RF = RandomForestClassifier().fit(x_train, y_train)
predictions_RF_test=clf_RF.predict(x_test)
print("Test Accuracy score of Random Forest after applying Smote is " +
str(round(accuracy_score(y_test, predictions_RF_test), 4)))
cv_predicted_test = cross_val_predict(clf_RF, x_test, y_test, cv=10)
print("Test Accuracy score of Random Forest with 10 folds cross
validation after applying Smote is " +
str(round(metrics.accuracy_score(y_test, cv_predicted_test),4)))
predictions_RF_train=clf_RF.predict(x_train)
print("Train Accuracy score of Random Forest after applying Smote is " +
str(round(accuracy_score(y_train, predictions_RF_train), 4)))
cv_predicted_train = cross_val_predict(clf_RF, x_train, y_train, cv=10)
print("Train Accuracy score of Random Forest with 10 folds cross
validation after applying Smote is " +
str(round(metrics.accuracy_score(y_train, cv_predicted_train),4)))
#Grid Search for improving the performance of Multinomial Naive Bayes
sorted(clf_NB.get_params().keys()) #displaying the all possible
parameters of Multinomial Naive Bayes
parameters_NB = {
    'fit_prior': ('true', 'false'),
    'alpha': [0.001, 0.0001, 0.00001, 0.000001]
}
gs_clf = GridSearchCV(clf_NB, parameters_NB, n_jobs=-1,
scoring='accuracy')
gs_clf = gs_clf.fit(x_train, y_train)
predictions_gs=gs_clf.predict(x_test)
print("The best score of Multinomial Naive Bayes with Grid Search after
applying Smote is " +  str(round(gs_clf.best_score_,4)))
print("Best parameters set found on Multinomial Naive Bayes algoritm:")
print(gs_clf.best_params_)
print( "Test Accuracy score of Multinomial Naive Bayes with Grid Search
after applying Smote is " +
str(round(accuracy_score(y_test,predictions_gs),4)))
predictions_gs_train=gs_clf.predict(x_train)
print( "Train Accuracy score of Multinomial Naive Bayes with Grid Search
after applying Smote is " +
str(round(accuracy_score(y_train,predictions_gs_train),4)))
cv_predicted_train = cross_val_predict(gs_clf, x_train, y_train, cv=10)
print("Train Accuracy score of Multinomial Naive Bayes with Grid Search
and 10 folds cross validation after applying Smote is " +
str(round(metrics.accuracy_score(y_train, cv_predicted_train),4)))
cv_predicted_test = cross_val_predict(gs_clf, x_test, y_test, cv=10)
print("Test Accuracy score of Multinomial Naive Bayes with Grid Search
and 10 folds cross validation after applying Smote is " +
str(round(metrics.accuracy_score(y_test, cv_predicted_test),4)))
#Confusion Matrix
labels = ['Kariyerim Bankacılık Üssü', 'Kariyerim Genel Müdürlük',
          'Kariyerim Şube','Kurumsal İK Yönetimi/Bordro İşlemleri',
          'Kurumsal İK Yönetimi/Özlük İşlemleri','Kurumsal İK
Yönetimi/Ücret ve Yan Haklar',
          'Yapı Kredi Bankacılık Akademisi','İşe Alım ve İK Geliştirme',
          ]
rf_cm = confusion_matrix(y_test, predictions_gs)
rf_cm_plot = pd.DataFrame(rf_cm)
fig = plt.figure(figsize = (8,6))
sns.heatmap(rf_cm_plot, annot=True, vmin=5, vmax=50.5, cbar=False,
fmt='g')
ax = fig.add_subplot(111)
cax = ax.matshow(rf_cm)
ax.set_xticklabels([''] + labels, rotation=45, fontsize=8 )
ax.set_yticklabels([''] + labels,  rotation=30, fontsize=8 )
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
#Classification Report
print(classification_report(y_test, predictions_gs, target_names=labels))
#Precision-Recall Curve
y_score = gs_clf.predict_proba(x_test)
rowCount = x_test.shape[0]
actualArr =  np.zeros((rowCount, len(labels)))
i=0
for y in y_test:
    actualArr[i][y] =1
    i=i+1
from sklearn.preprocessing import label_binarize
# Use label_binarize to be multi-label like settings
Y = label_binarize(df_y, classes=[0, 1, 2, 3, 4, 5, 6, 7])
n_classes = Y.shape[1]
print(n_classes)
y_test_arr=np.array(y_test)
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import average_precision_score
precision = dict()
recall = dict()
average_precision = dict()
for i in range(n_classes):
    precision[i], recall[i], _ = precision_recall_curve(actualArr[:, i],
                                                        y_score[:, i])
    average_precision[i] = average_precision_score(actualArr[:, i],
y_score[:, i])
# A "micro-average": quantifying score on all classes jointly
precision["micro"], recall["micro"], _ =
precision_recall_curve(actualArr.ravel(),

y_score.ravel())
average_precision["micro"] = average_precision_score(actualArr, y_score,
                                                     average="micro")
from itertools import cycle
# setup plot details
colors = cycle(['navy', 'turquoise', 'darkorange', 'cornflowerblue',
'teal'])
plt.figure(figsize=(7, 8))
f_scores = np.linspace(0.2, 0.8, num=4)
lines = []
labels = []
for f_score in f_scores:
    x = np.linspace(0.01, 1)
    y = f_score * x / (2 * x - f_score)
    l, = plt.plot(x[y >= 0], y[y >= 0], color='gray', alpha=0.2)
    plt.annotate('f1={0:0.1f}'.format(f_score), xy=(0.9, y[45] + 0.02))
lines.append(l)
labels.append('iso-f1 curves')
l, = plt.plot(recall["micro"], precision["micro"], color='gold', lw=2)
lines.append(l)
labels.append('micro-average Precision-recall (area = {0:0.2f})'
              ''.format(average_precision["micro"]))
departmentName=''
for i, color in zip(range(n_classes), colors):
    if i == 0:
        departmentName='Kariyerim Bankacılık Üssü'
    elif i == 1:
        departmentName='Kariyerim Bankacılık Üssü'
    elif i == 2:
        departmentName='Kariyerim Şube'
    elif i == 3:
        departmentName='Kurumsal İK Yönetimi/Bordro İşlemleri'
    elif i == 4:
        departmentName='Kurumsal İK Yönetimi/Özlük İşlemleri'
    elif i == 5:
        departmentName='Kurumsal İK Yönetimi/Ücret ve Yan Haklar'
    elif i == 6:
        departmentName='Yapı Kredi Bankacılık Akademisi'
    else:
        departmentName='İşe Alım ve İK Geliştirme'
    l, = plt.plot(recall[i], precision[i], color=color, lw=2)
    lines.append(l)
    labels.append('Precision-recall for class {0} (area = {1:0.2f})'
                  ''.format(departmentName, average_precision[i]))
fig = plt.gcf()
fig.subplots_adjust(bottom=0.25)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title(' Precision-Recall Curve')
plt.legend(lines, labels, loc=(0, -.38), prop=dict(size=14))
plt.show()
```