Match Prediction

BBM480 Design Project

Advisor :

Asst. Prof. Dr. Burkay Genç

Students:

İsmail Ateş 21626953

Mustafa Kollu 21627485

Emre Hancı 21604552

Summary

Our project is about collecting data about football matches and training machine learning model for result prediction and create a platform(website) to users can be see the prediction of trained model. The model of prediction football matches will evaluate for reliability of prediction, improving predictability with kind of criteria such that pass matches between given two teams.

PRODUCT FEATURES

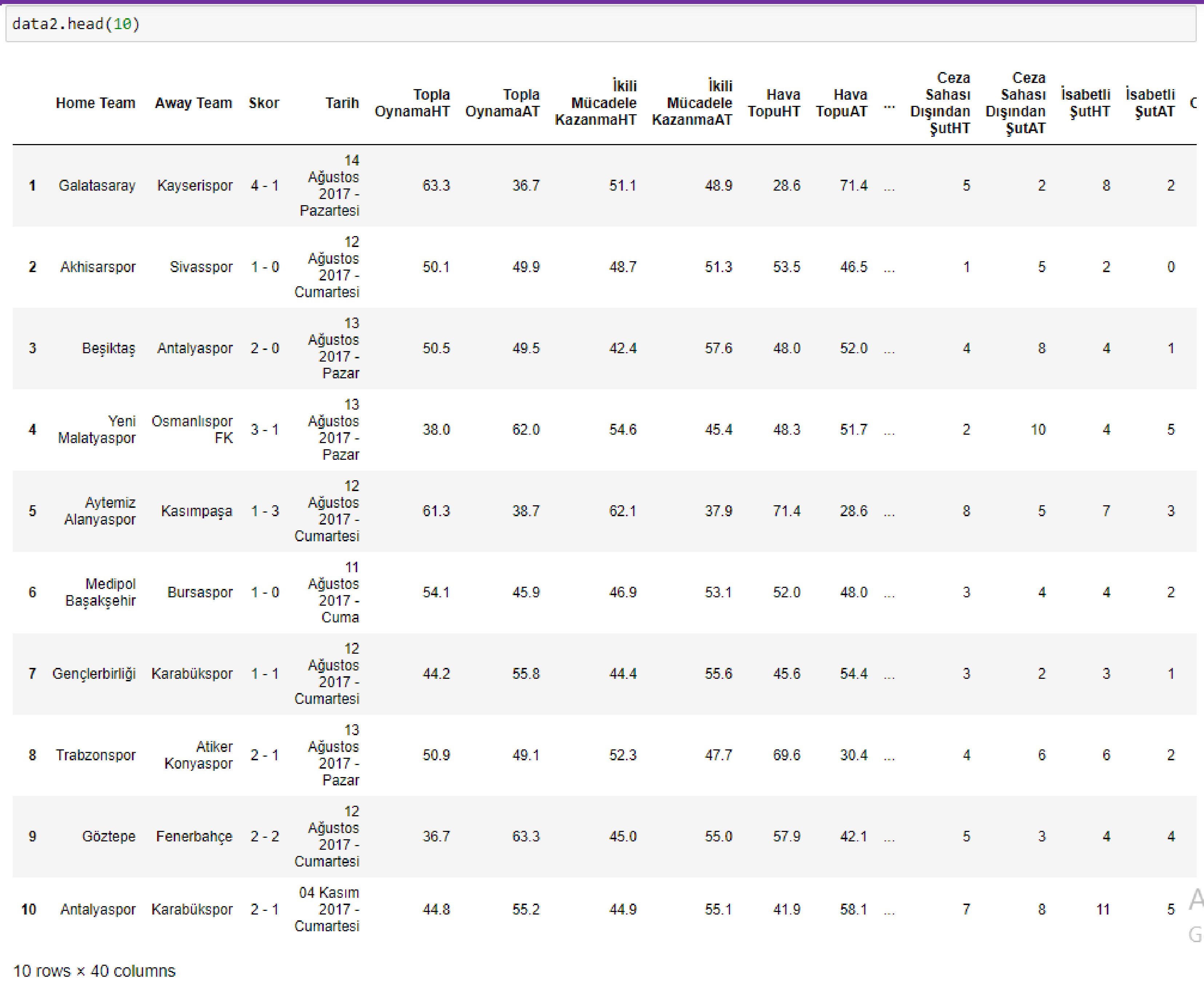
1) Collect Data
2) Data understanding
3) Data preprocess
4) Training model
5) Evaluating model
6) Creating platform to show prediction of matches.
7) Prediction of the next matches / Given two team matches

Collect Data

2017-2018 and 2018-2019 Turkey 1st League data were obtained with web scraping. The tool developed by Professor Burkay was used for web scraping. Afterwards, these obtained data are stored in a web api to be used in the model development phase. This web api is written by us. An example part of the web api is as follows.

Data understanding

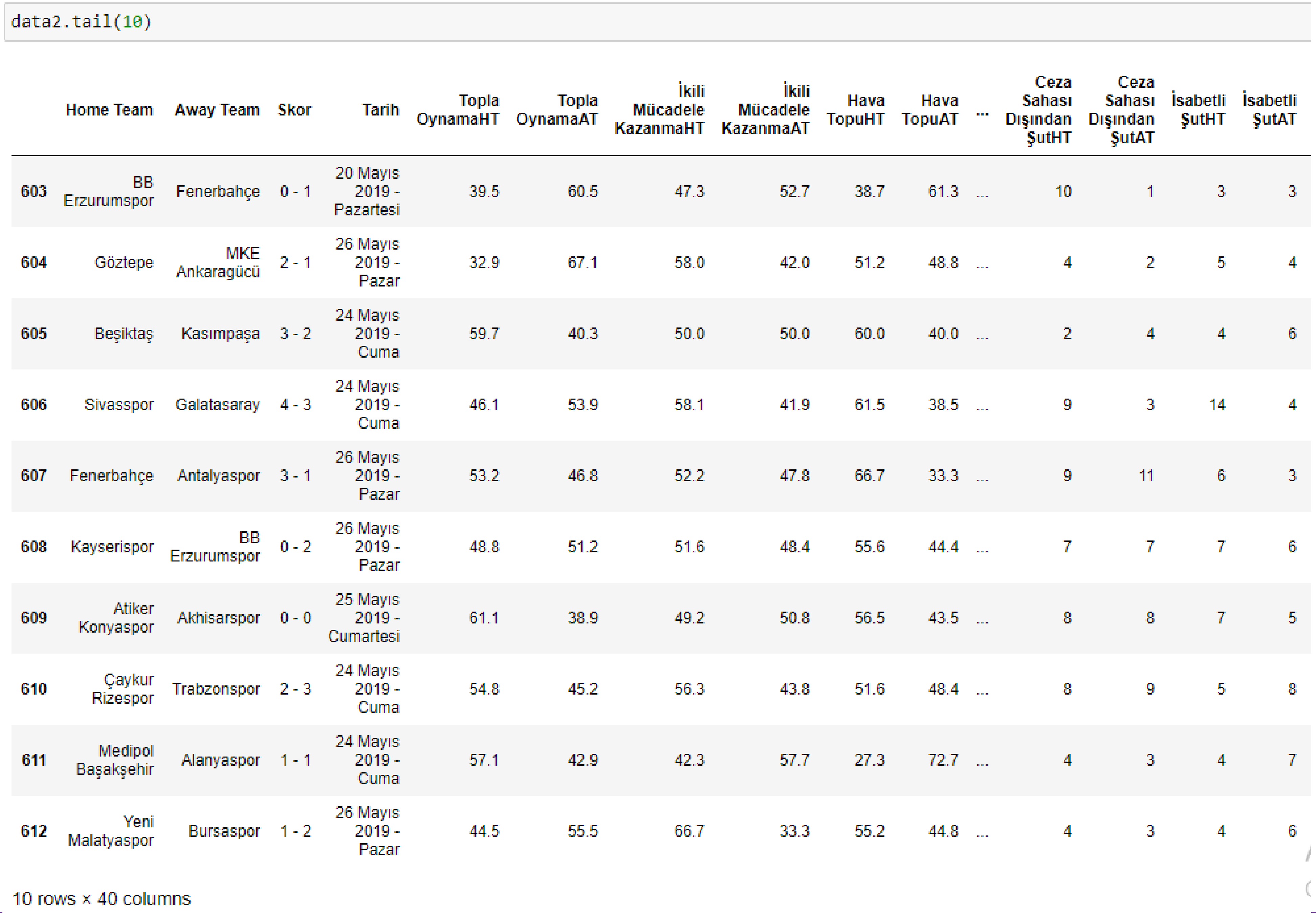
Understanding data 'Is the data collected sufficient to solve our problem?' replies to the question. We examined the rows and columns of our data at the stage of understanding the data. We will draw graphs like histograms and interpret our data set for a better understanding of the data. With such methodologies, we will observe in detail whether there is missing data, duplicate data and invalid data in our data. We start the process of understanding the data by reading our data.Below we see the first and last 10 rows of our data set. We see that the value ranges of some of our data are very low.



```
data2.head(10)
```

	Home Team	Away Team	Skor	Tarih	Topla OynamaHT	Topla OynamaAT	İkili Mücadele KazanmaHT	İkili Mücadele KazanmaAT	Hava TopuHT	Hava TopuAT	...	Ceza Sahası Dışından ŞutHT	Ceza Sahası Dışından ŞutAT	İsabetli ŞutHT	İsabetli ŞutAT
1	Galatasaray	Kayserispor	4 - 1	14 Ağustos 2017 - Pazartesi	63.3	36.7	51.1	48.9	28.6	71.4	...	5	2	8	2
2	Akhisarspor	Sivasspor	1 - 0	12 Ağustos 2017 - Cumartesi	50.1	49.9	48.7	51.3	53.5	46.5	...	1	5	2	0
3	Beşiktaş	Antalyaspor	2 - 0	13 Ağustos 2017 - Pazar	50.5	49.5	42.4	57.6	48.0	52.0	...	4	8	4	1
4	Yeni Malatyaspor	Osmanlıspor FK	3 - 1	13 Ağustos 2017 - Pazar	38.0	62.0	54.6	45.4	48.3	51.7	...	2	10	4	5
5	Aytemiz Alanyaspor	Kasımpaşa	1 - 3	12 Ağustos 2017 - Cumartesi	61.3	38.7	62.1	37.9	71.4	28.6	...	8	5	7	3
6	Medipol Başakşehir	Bursaspor	1 - 0	11 Ağustos 2017 - Cuma	54.1	45.9	46.9	53.1	52.0	48.0	...	3	4	4	2
7	Gençlerbirliği	Karabükspor	1 - 1	12 Ağustos 2017 - Cumartesi	44.2	55.8	44.4	55.6	45.6	54.4	...	3	2	3	1
8	Trabzonspor	Atiker Konyaspor	2 - 1	13 Ağustos 2017 - Pazar	50.9	49.1	52.3	47.7	69.6	30.4	...	4	6	6	2
9	Göztepe	Fenerbahçe	2 - 2	12 Ağustos 2017 - Cumartesi	36.7	63.3	45.0	55.0	57.9	42.1	...	5	3	4	4
10	Antalyaspor	Karabükspor	2 - 1	04 Kasım 2017 - Cumartesi	44.8	55.2	44.9	55.1	41.9	58.1	...	7	8	11	5

10 rows × 40 columns

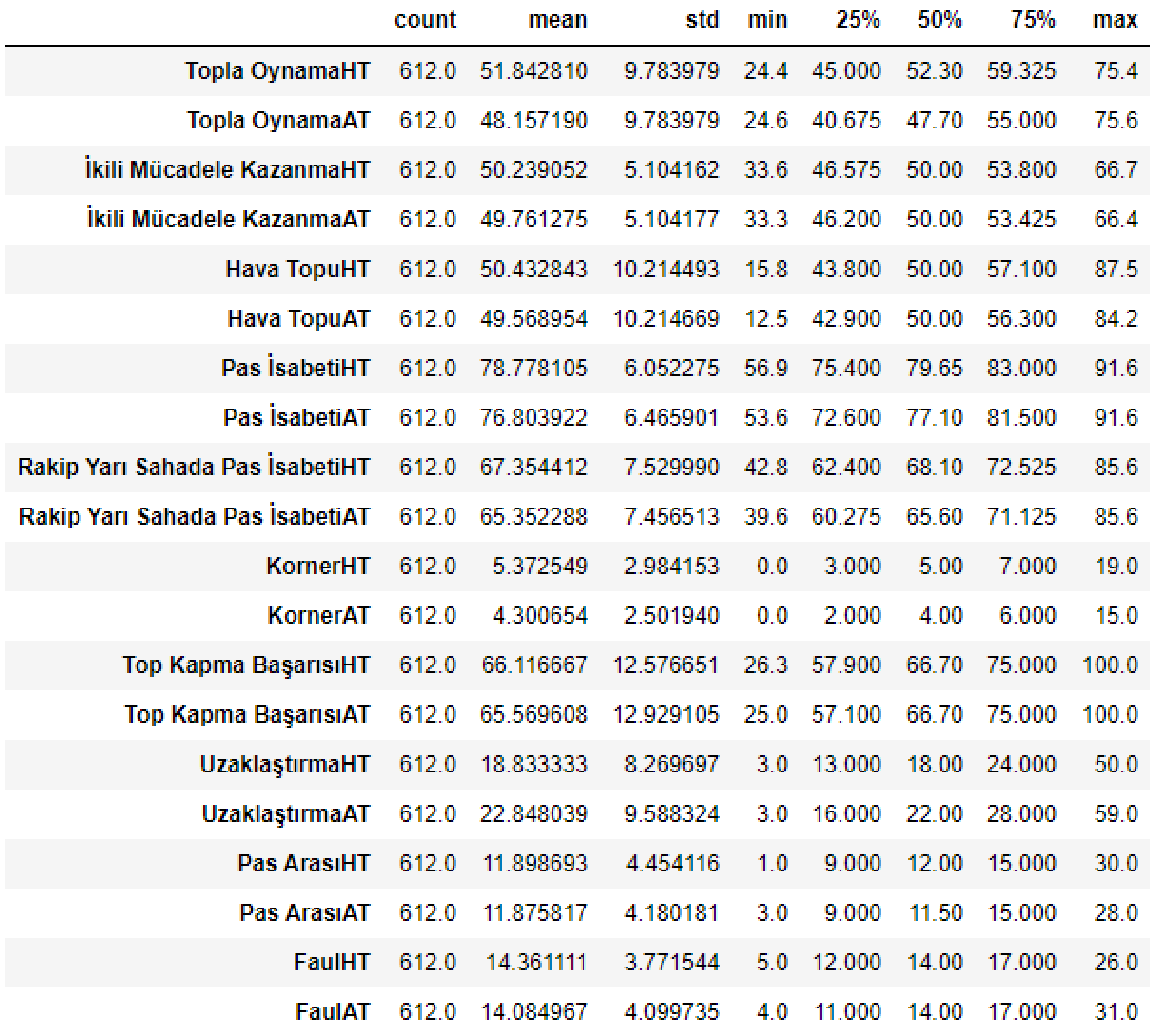
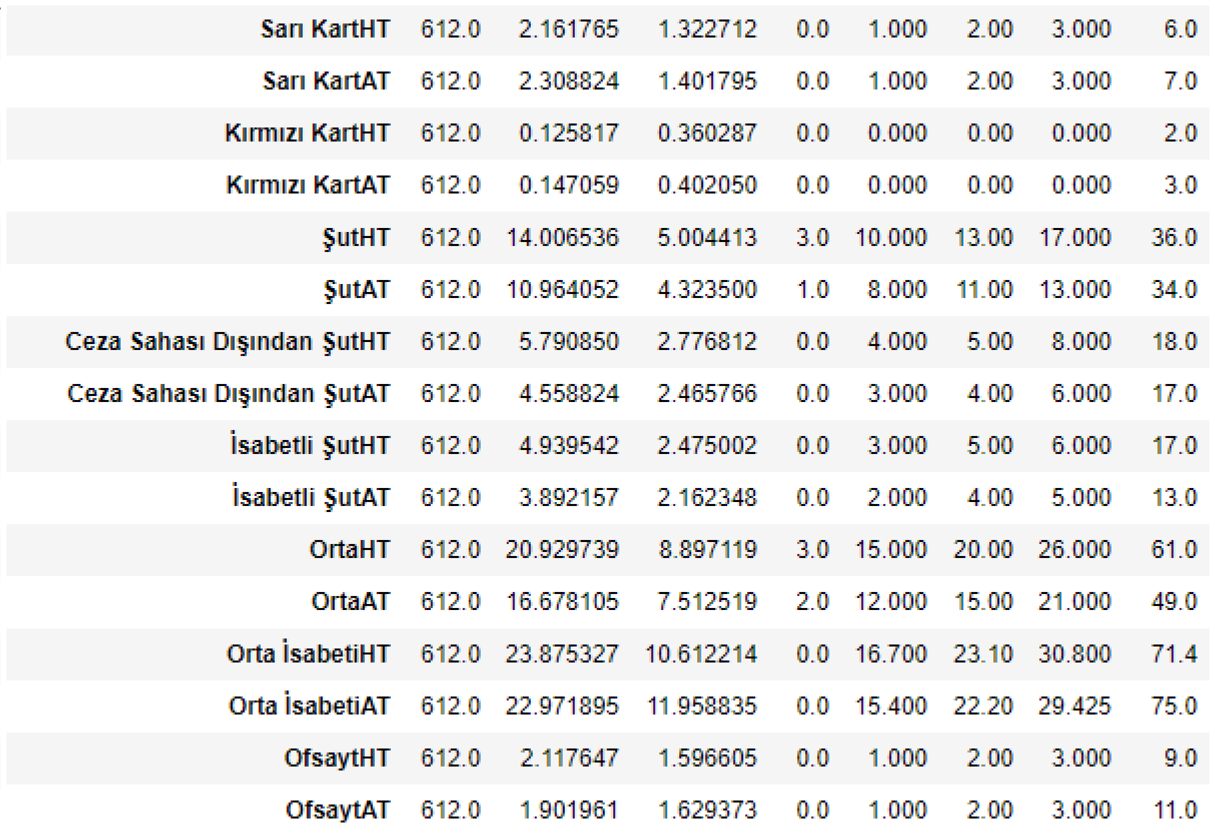


```
data2.tail(10)
```

	Home Team	Away Team	Skor	Tarih	Topla OynamaHT	Topla OynamaAT	İkili Mücadele KazanmaHT	İkili Mücadele KazanmaAT	Hava TopuHT	Hava TopuAT	...	Ceza Sahası Dışından ŞutHT	Ceza Sahası Dışından ŞutAT	İsabetli ŞutHT	İsabetli ŞutAT
603	BB Erzurumspor	Fenerbahçe	0 - 1	20 Mayıs 2019 - Pazartesi	39.5	60.5	47.3	52.7	38.7	61.3	...	10	1	3	3
604	Göztepe	MKE Ankaragücü	2 - 1	26 Mayıs 2019 - Pazar	32.9	67.1	58.0	42.0	51.2	48.8	...	4	2	5	4
605	Beşiktaş	Kasımpaşa	3 - 2	24 Mayıs 2019 - Cuma	59.7	40.3	50.0	50.0	60.0	40.0	...	2	4	4	6
606	Sivasspor	Galatasaray	4 - 3	24 Mayıs 2019 - Cuma	46.1	53.9	58.1	41.9	61.5	38.5	...	9	3	14	4
607	Fenerbahçe	Antalyaspor	3 - 1	26 Mayıs 2019 - Pazar	53.2	46.8	52.2	47.8	66.7	33.3	...	9	11	6	3
608	Kayserispor	BB Erzurumspor	0 - 2	26 Mayıs 2019 - Pazar	48.8	51.2	51.6	48.4	55.6	44.4	...	7	7	7	6
609	Atiker Konyaspor	Akhisarspor	0 - 0	25 Mayıs 2019 - Cumartesi	61.1	38.9	49.2	50.8	56.5	43.5	...	8	8	7	5
610	Çaykur Rizespor	Trabzonspor	2 - 3	24 Mayıs 2019 - Cuma	54.8	45.2	56.3	43.8	51.6	48.4	...	8	9	5	8
611	Medipol Başakşehir	Alanyaspor	1 - 1	24 Mayıs 2019 - Cuma	57.1	42.9	42.3	57.7	27.3	72.7	...	4	3	4	7
612	Yeni Malatyaspor	Bursaspor	1 - 2	26 Mayıs 2019 - Pazar	44.5	55.5	66.7	33.3	55.2	44.8	...	4	3	4	6

10 rows × 40 columns

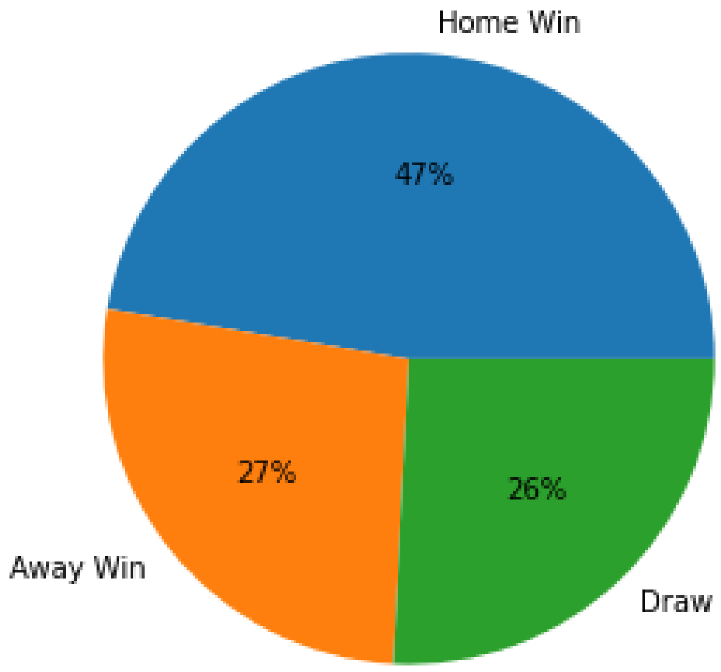
In the data information section, we see that some of our data are floats and some are strings. Since String data other than the target column will not affect our model, we will discard them at the data preprocess stage. We will also encode the target column. At the same time, we see that our dataset consists of 612 rows and 40 columns.
You can find statistical information about our data set in the table below. Here we see values such as mean, standard deviation, min-max, number of data, lower quartile-upper quartile and median for each column.

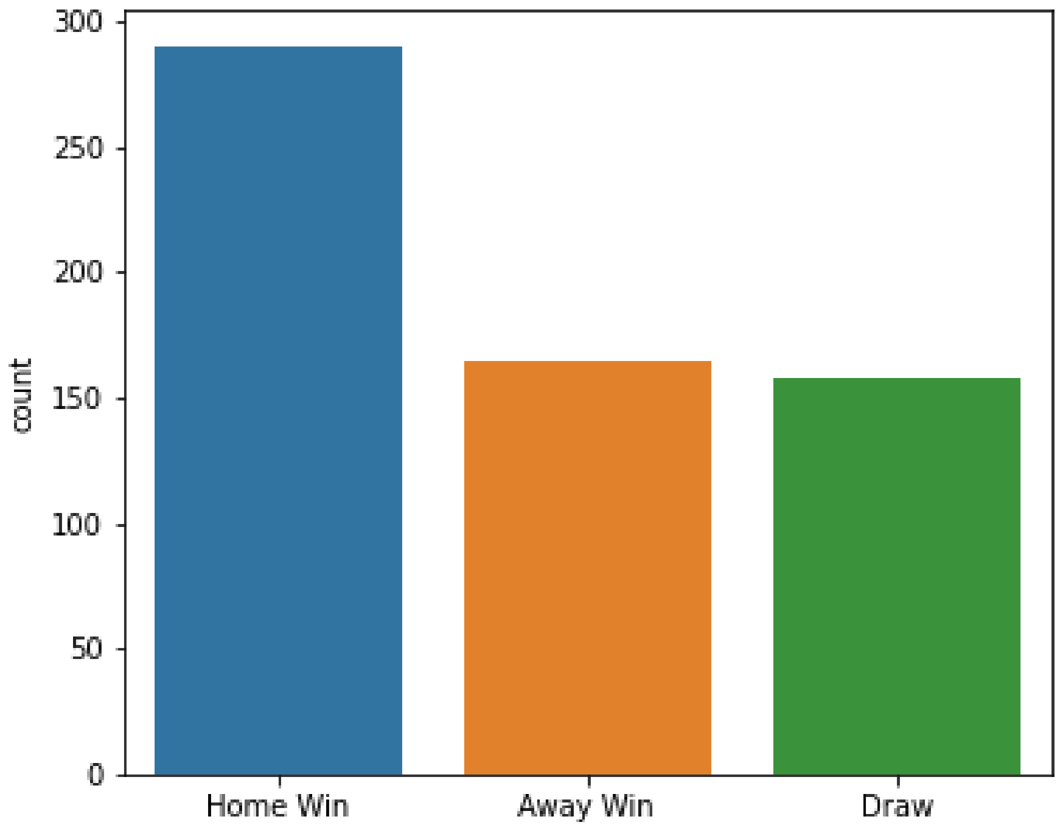



	count	mean	std	min	25%	50%	75%	max
Topla OynamaHT	612.0	51.842810	9.783979	24.4	45.000	52.30	59.325	75.4
Topla OynamaAT	612.0	48.157190	9.783979	24.6	40.675	47.70	55.000	75.6
İkili Mücadele KazanmaHT	612.0	50.239052	5.104162	33.6	46.575	50.00	53.800	66.7
İkili Mücadele KazanmaAT	612.0	49.761275	5.104177	33.3	46.200	50.00	53.425	66.4
Hava TopuHT	612.0	50.432843	10.214493	15.8	43.800	50.00	57.100	87.5
Hava TopuAT	612.0	49.568954	10.214669	12.5	42.900	50.00	56.300	84.2
Pas İsabetiHT	612.0	78.778105	6.052275	56.9	75.400	79.65	83.000	91.6
Pas İsabetiAT	612.0	76.803922	6.465901	53.6	72.600	77.10	81.500	91.6
Rakip Yarı Sahada Pas İsabetiHT	612.0	67.354412	7.529990	42.8	62.400	68.10	72.525	85.6
Rakip Yarı Sahada Pas İsabetiAT	612.0	65.352288	7.456513	39.6	60.275	65.60	71.125	85.6
KornerHT	612.0	5.372549	2.984153	0.0	3.000	5.00	7.000	19.0
KornerAT	612.0	4.300654	2.501940	0.0	2.000	4.00	6.000	15.0
Top Kapma BaşarısıHT	612.0	66.116667	12.576651	26.3	57.900	66.70	75.000	100.0
Top Kapma BaşarısıAT	612.0	65.569608	12.929105	25.0	57.100	66.70	75.000	100.0
UzaklaştırmaHT	612.0	18.833333	8.269697	3.0	13.000	18.00	24.000	50.0
UzaklaştırmaAT	612.0	22.848039	9.588324	3.0	16.000	22.00	28.000	59.0
Pas ArasıHT	612.0	11.898693	4.454116	1.0	9.000	12.00	15.000	30.0
Pas ArasıAT	612.0	11.875817	4.180181	3.0	9.000	11.50	15.000	28.0
FaulHT	612.0	14.361111	3.771544	5.0	12.000	14.00	17.000	26.0
FaulAT	612.0	14.084967	4.099735	4.0	11.000	14.00	17.000	31.0
Sarı KartHT	612.0	2.161765	1.322712	0.0	1.000	2.00	3.000	6.0
Sarı KartAT	612.0	2.308824	1.401795	0.0	1.000	2.00	3.000	7.0
Kırmızı KartHT	612.0	0.125817	0.360287	0.0	0.000	0.00	0.000	2.0
Kırmızı KartAT	612.0	0.147059	0.402050	0.0	0.000	0.00	0.000	3.0
ŞutHT	612.0	14.006536	5.004413	3.0	10.000	13.00	17.000	36.0
ŞutAT	612.0	10.964052	4.323500	1.0	8.000	11.00	13.000	34.0
Ceza Sahası Dışından ŞutHT	612.0	5.790850	2.776812	0.0	4.000	5.00	8.000	18.0
Ceza Sahası Dışından ŞutAT	612.0	4.558824	2.465766	0.0	3.000	4.00	6.000	17.0
İsabetli ŞutHT	612.0	4.939542	2.475002	0.0	3.000	5.00	6.000	17.0
İsabetli ŞutAT	612.0	3.892157	2.162348	0.0	2.000	4.00	5.000	13.0
OrtaHT	612.0	20.929739	8.897119	3.0	15.000	20.00	26.000	61.0
OrtaAT	612.0	16.678105	7.512519	2.0	12.000	15.00	21.000	49.0
Orta İsabetiHT	612.0	23.875327	10.612214	0.0	16.700	23.10	30.800	71.4
Orta İsabetiAT	612.0	22.971895	11.958835	0.0	15.400	22.20	29.425	75.0
OfsaytHT	612.0	2.117647	1.596605	0.0	1.000	2.00	3.000	9.0
OfsaytAT	612.0	1.901961	1.629373	0.0	1.000	2.00	3.000	11.0

Target Column Review

When we look at the 'score' column, which is our target tag, we see that the majority of the match scores are home wins. However, we also see that the match





Data Preparation

In the data preparation phase, there are all the activities that create the data to be used for modeling. In the data understanding part of our project, we observed that our data is not very complex. There were problems in some places, such as column capitals. In the data preparation phase, we will clean our data, check the missing and duplicate data, and adapt our data to machine learning approaches.

1. Why do we do data preparation?

Simply put, it is the process of preparing data, getting raw data and making it available for use in an analytics platform. To reach the final stage of preparation, the data must be cleaned, formatted and turned into something digestible with analysis tools. These can include a wide variety of steps, such as merging / splitting columns, changing formats, deleting unnecessary or unnecessary data, and making corrections to the data.

2. What does all this time and effort do for data scientists?

It's all about trust. Trust in data, trust in the process, and trust in insights drawn from the data. Data preparation ensures accuracy in data, which provides accurate insights. Without data preparation, insights may be closed due to trivial data, an overlooked calibration issue, or an easily corrected discrepancy between data sets.

We discard the date and team names from our dataset to make our dataset fit the model. We make the target column understandable.

```
data = data2.drop(columns = ["Home Team","Away Team", 'Tarih'])
```

```
data[['ScoreHT','ScoreAT']] = data.Skor.str.split(" - ",expand=True,)
```

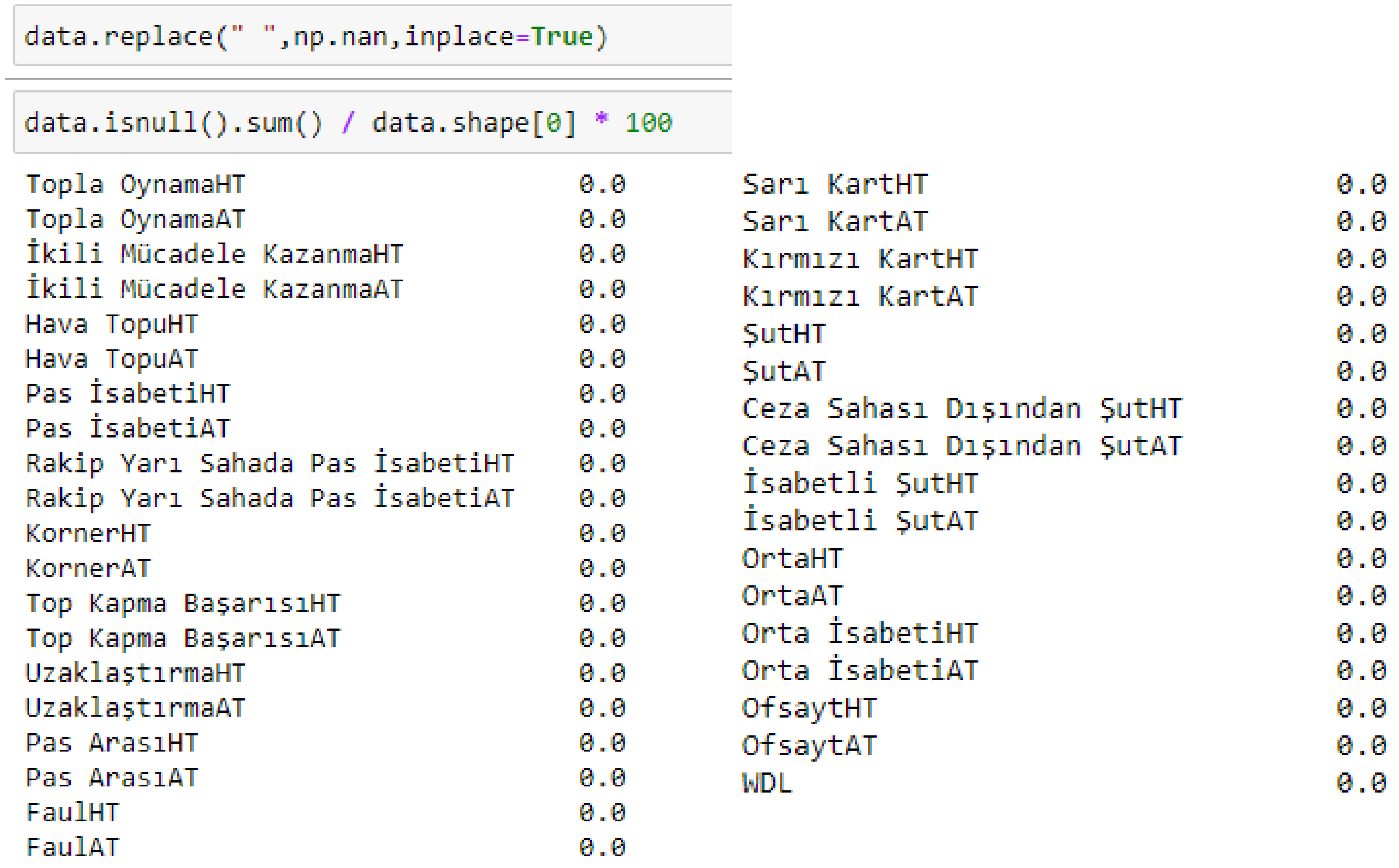
```
data["ScoreHT"] = data["ScoreHT"].apply(pd.to_numeric)
data["ScoreAT"] = data["ScoreAT"].apply(pd.to_numeric)
```

```
data["WDL"] = data["ScoreHT"]-data["ScoreAT"]
```

```
def encode(x):
    if x > 0:
        return 1
    elif x < 0:
        return 2
    else:
        return 0

data["WDL"] = data.apply(lambda x: encode(x["WDL"]), axis = 1)
data["WDL"]
```

```
1      1
2      1
3      1
4      1
5      2
      ..
608    2
609    0
610    2
611    0
612    2
Name: WDL, Length: 612, dtype: int64
```

First of all, we convert the null values (if any) in our data to NaN values. After this process, we check the NaN values. We see that there are no spaces and no NaN values in our data set.

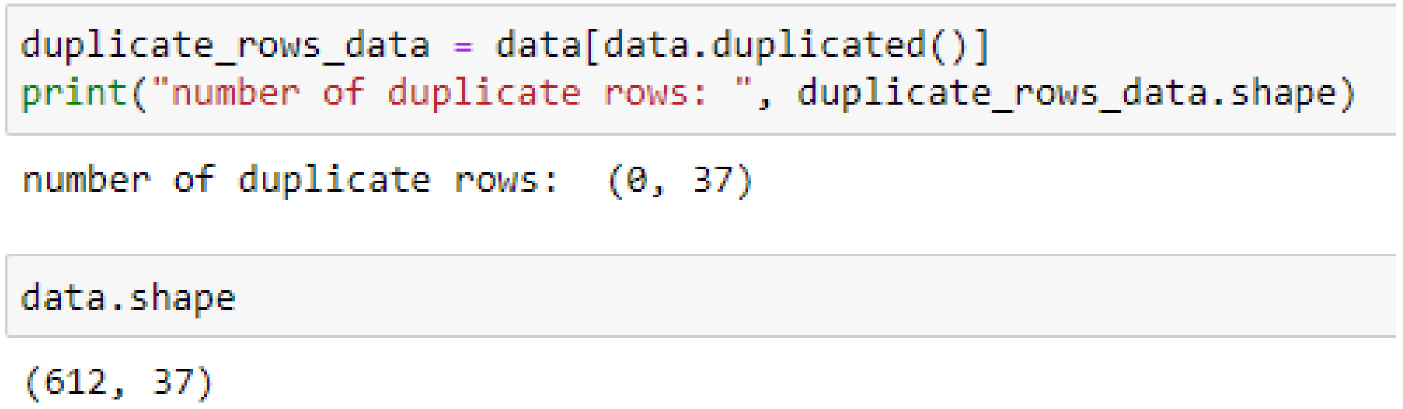


```
data.replace(" ",np.nan,inplace=True)
```

```
data.isnull().sum() / data.shape[0] * 100
```

```
Topla OynamaHT                        0.0
Topla OynamaAT                        0.0
İkili Mücadele KazanmaHT              0.0
İkili Mücadele KazanmaAT              0.0
Hava TopuHT                           0.0
Hava TopuAT                           0.0
Pas İsabetiHT                         0.0
Pas İsabetiAT                         0.0
Rakip Yarı Sahada Pas İsabetiHT       0.0
Rakip Yarı Sahada Pas İsabetiAT       0.0
KornerHT                              0.0
KornerAT                              0.0
Top Kapma BaşarısıHT                  0.0
Top Kapma BaşarısıAT                  0.0
UzaklaştırmaHT                        0.0
UzaklaştırmaAT                        0.0
Pas ArasıHT                           0.0
Pas ArasıAT                           0.0
FaulHT                                0.0
FaulAT                                0.0
Sarı KartHT                           0.0
Sarı KartAT                           0.0
Kırmızı KartHT                        0.0
Kırmızı KartAT                        0.0
ŞutHT                                 0.0
ŞutAT                                 0.0
Ceza Sahası Dışından ŞutHT            0.0
Ceza Sahası Dışından ŞutAT            0.0
İsabetli ŞutHT                        0.0
İsabetli ŞutAT                        0.0
OrtaHT                                0.0
OrtaAT                                0.0
Orta İsabetiHT                        0.0
Orta İsabetiAT                        0.0
OfsaytHT                              0.0
OfsaytAT                              0.0
WDL                                   0.0
```

After checking whether there are gap-NaN values in our data set, we check if there are duplicate rows in our data. We see that there are no duplicate rows in our dataset.



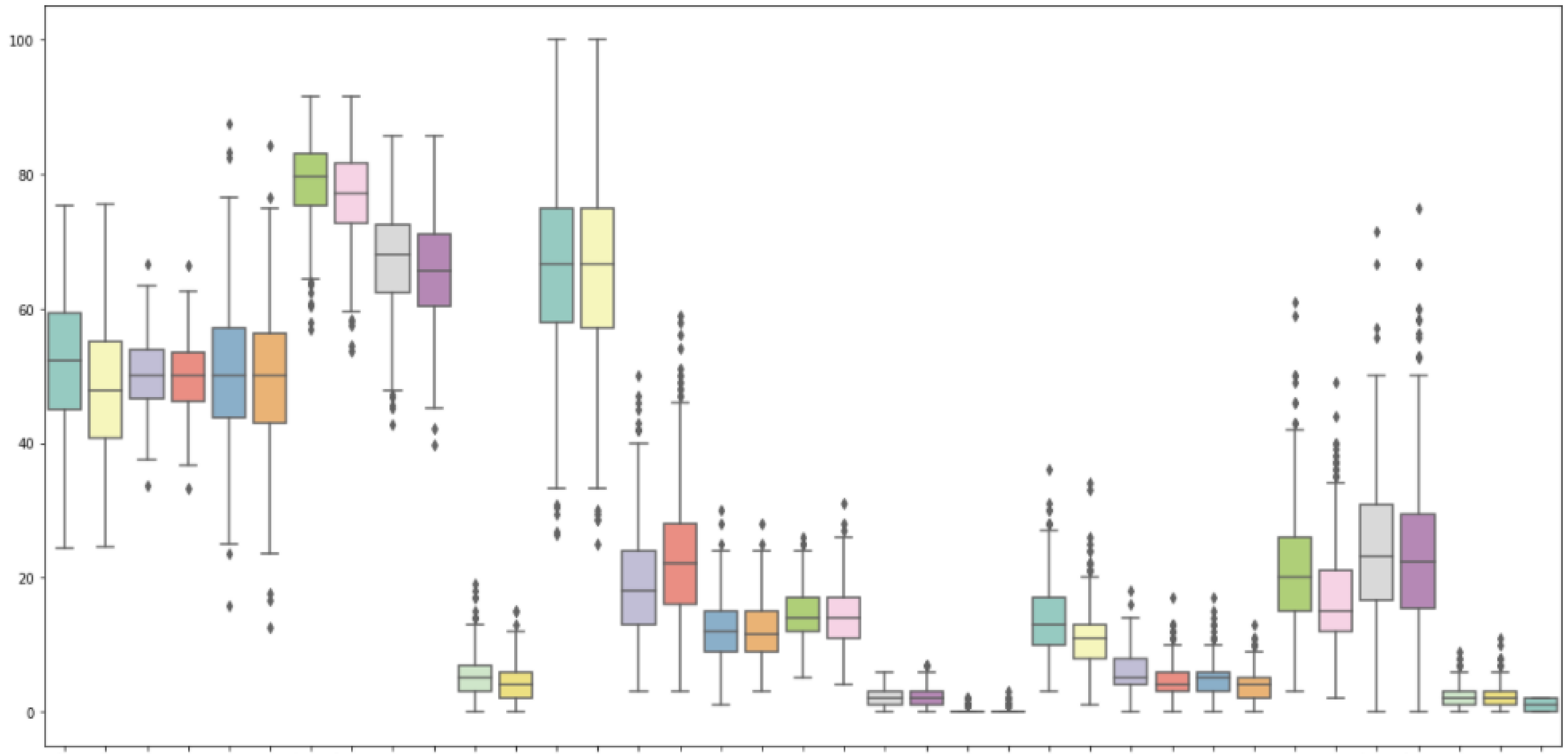
```
duplicate_rows_data = data[data.duplicated()]
print("number of duplicate rows: ", duplicate_rows_data.shape)
```

```
number of duplicate rows:  (0, 37)
```

```
data.shape
```

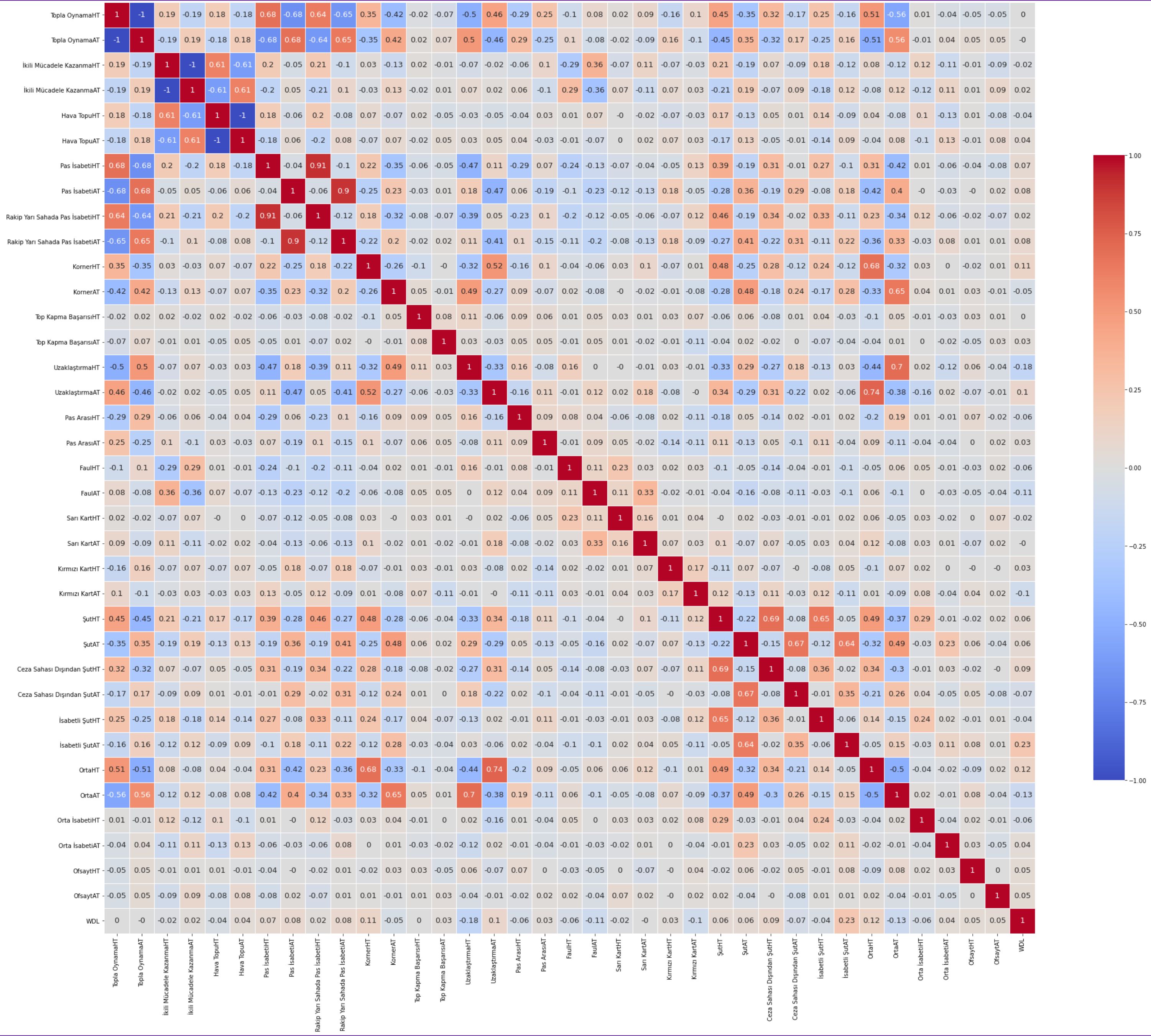
```
(612, 37)
```

When we look at the outliers in the data set, we did not interfere with the data set because we thought that outliers would not cause a problem.



In statistics, correlation is any statistical relationship, whether causal or not, between two random variables or data with two variables. In the broadest sense, correlation is any statistical relationship, but generally refers to the degree to which a pair of variables is linearly related. Known examples of dependent phenomena include the correlation between parents' height and their children, and the correlation between the price of a good and the quantity consumers are willing to buy, as depicted in the so-called demand curve. We check if there is a relationship between the independent and target variables.

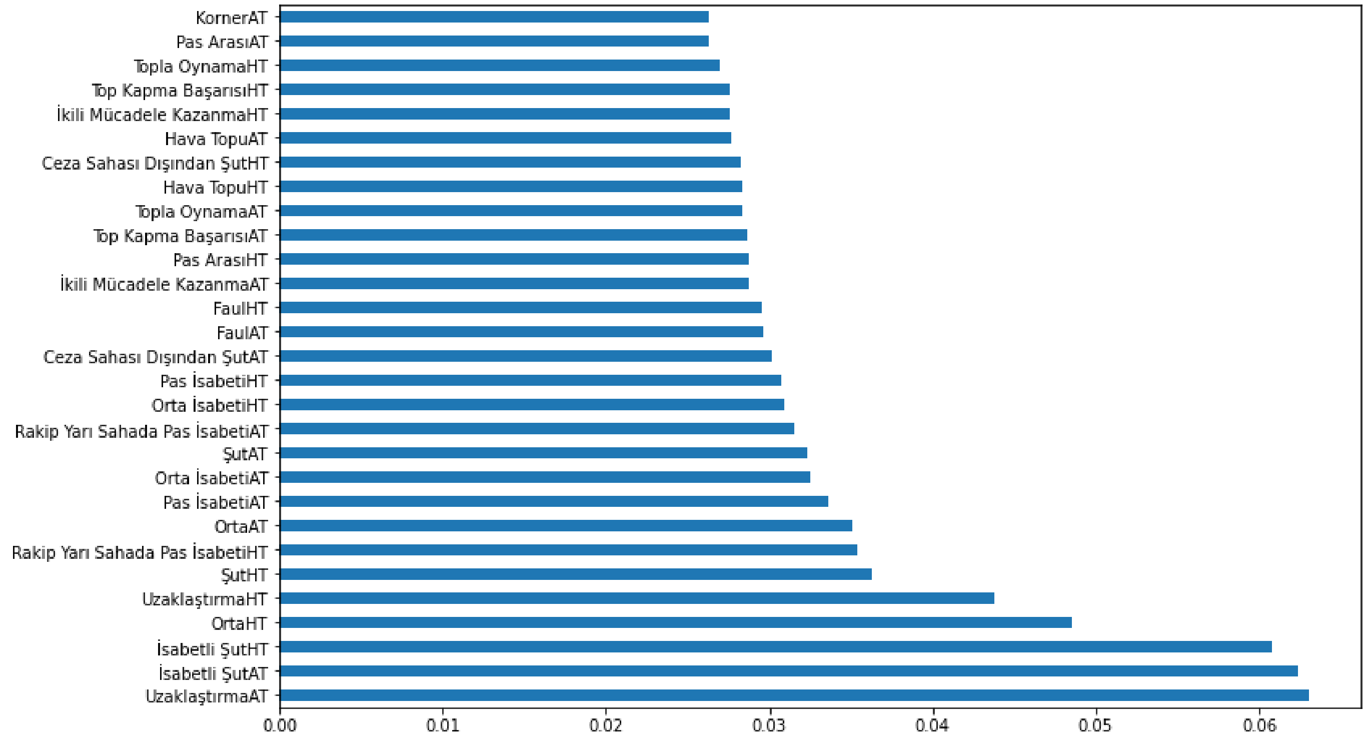
We checked the relationship between features and match statistics with data.corr() provided by the Python library Pandas. We draw a heatmap to examine the relationship between these values. When we looked at the relationship between match score and features, we saw that many features were unimportant. After these operations, we removed the columns with low correlation from our data set. When we look at the final version of the data set, we see that we have a total of 31 columns with the target column.



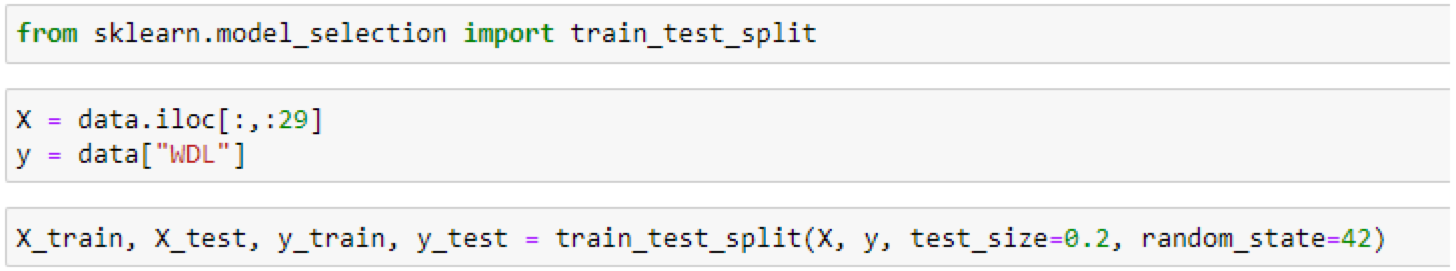
Dropping Columns with Low Correlation

```
data = data[['Topla OynamaHT', 'Topla OynamaAT', 'İkili Mücadele KazanmaHT',
        'İkili Mücadele KazanmaAT', 'Hava TopuHT', 'Hava TopuAT',
        'Pas İsabetiHT', 'Pas İsabetiAT', 'Rakip Yarı Sahada Pas İsabetiHT',
        'Rakip Yarı Sahada Pas İsabetiAT', 'KornerAT', 'Top Kapma BaşarısıHT',
        'Top Kapma BaşarısıAT', 'UzaklaştırmaHT', 'UzaklaştırmaAT',
        'Pas ArasıHT', 'Pas ArasıAT', 'FaulHT', 'FaulAT', 'ŞutHT', 'ŞutAT',
        'Ceza Sahası Dışından ŞutHT', 'Ceza Sahası Dışından ŞutAT',
        'İsabetli ŞutHT', 'İsabetli ŞutAT', 'OrtaHT', 'OrtaAT',
        'Orta İsabetiHT', 'Orta İsabetiAT',"WDL"]]
```

Since the purpose of this model we created is to determine the match result with less time and cost, we tried to reduce the number of features to the number required for our model. That's why we thought about how much we could reduce our feature count. The way to do this was to check which of the remaining features was more important to the model. We tried to test the importance of features with the ExtraTreesClassifier in the scikit-learn library.



As the last stage of data preparation, we separate our data set as train data and test data. After performing the separation process, we normalize the features of our train and test data. We removed this step because the accuracy of the model decreased when we used MinMaxScaler or Standardscaler.



```
from sklearn.model_selection import train_test_split
```

```
X = data.iloc[:,:29]
y = data["WDL"]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Modeling

In the previous stages, we analyzed and cleaned our data. In the modeling and evaluation phase, we will use many modeling techniques and train our model.

Supervised learning is a subcategory of machine learning and artificial intelligence. It is defined by using tagged datasets to train algorithms to classify data or accurately predict results. As the input data is fed into the model, it adjusts its weights until the model is properly fitted, and this happens as part of the cross validation process. Supervised learning helps organizations solve a variety of real-world problems at scale, such as categorizing spam in a separate folder from your inbox.

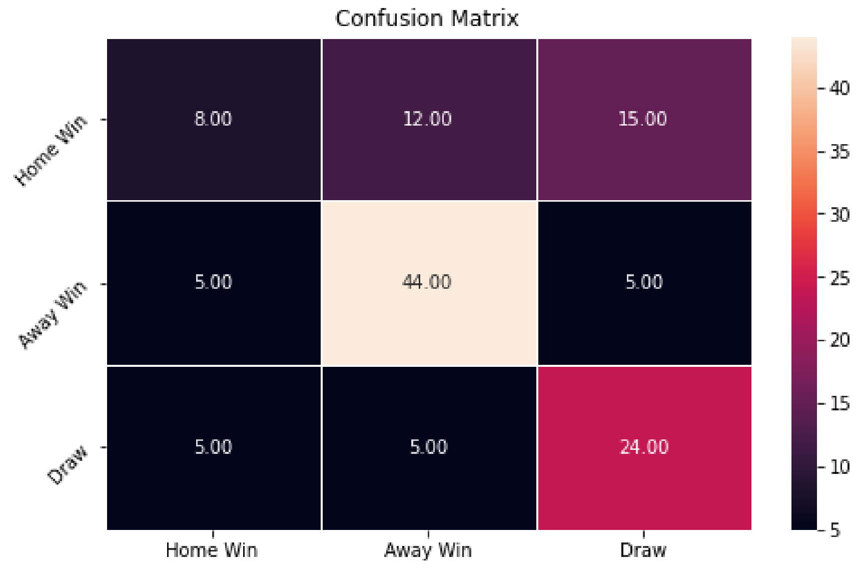
At this stage, we will use many supervised learning techniques and we will see the evaluation of these techniques. These techniques are Random Forest Classifier, K-Nearest Neighbors, Decision Tree Classifier, Support Vector Machine, Gaussian Naive Bayes.

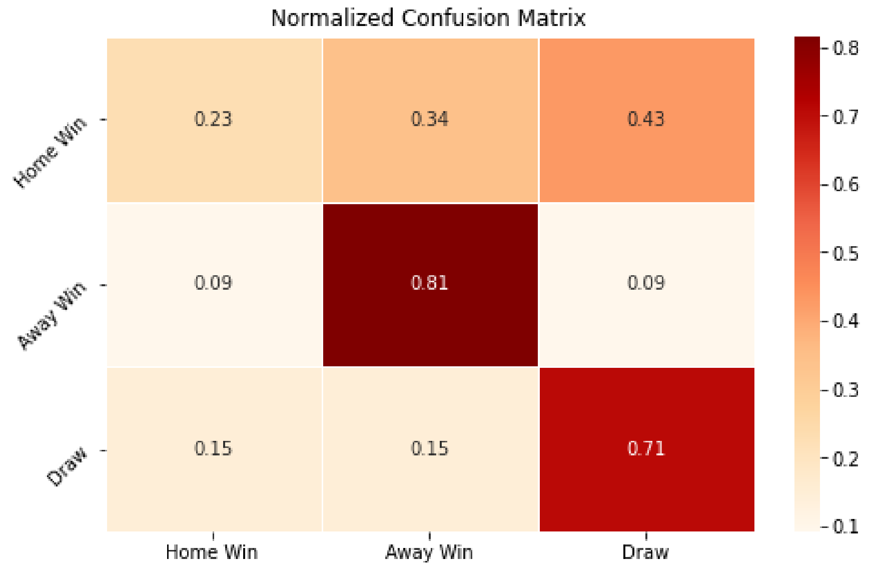
Random Forest Classifier

We decided to use this machine learning model because the random forest gives good results without hyperparameter estimation and can be applied to both regression and classification problems. The random forest can be compared to people who ask questions and make accurate predictions in daily life. But one of the biggest problems of decision trees, which is one of the traditional methods, is excessive learning-data memorization. In order to solve this problem, the random forest model randomly selects 10s and 100s of different subsets from both the data set and the feature set and trains them. With this method, hundreds of decision trees are created and each decision tree makes individual predictions.

At the end of the day, if our problem is regression, if our problem is classifying the average of the estimates of the decision trees, we choose the most votes among the predictions. Since we use the Cross-validation method in the Grid Search stage, we reduce the possibility of overfitting. Therefore, we will not have a problem using Random Forest.

Many hyperparameters are used in the Random Forest Classifier. We aimed to find the best hyperparameters in all the models we created using the hyperparameter adjustment method GridSearchCV method.

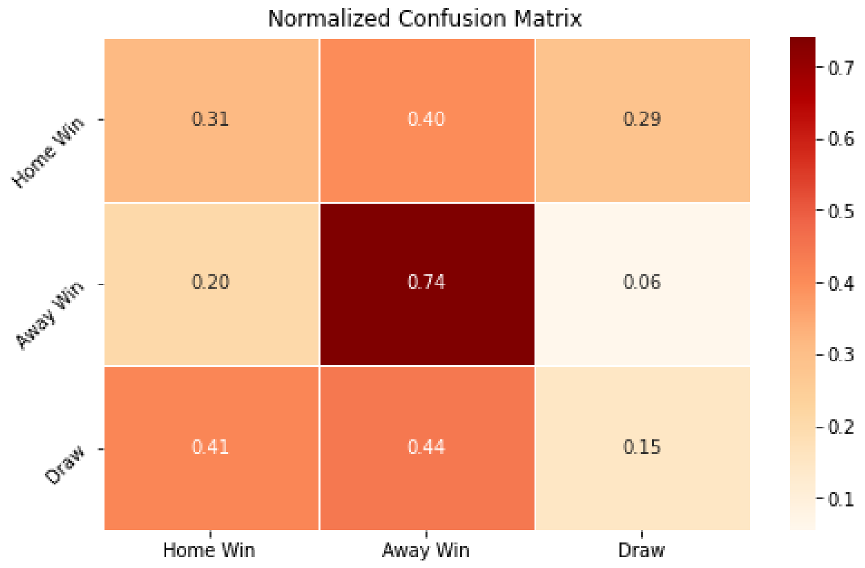


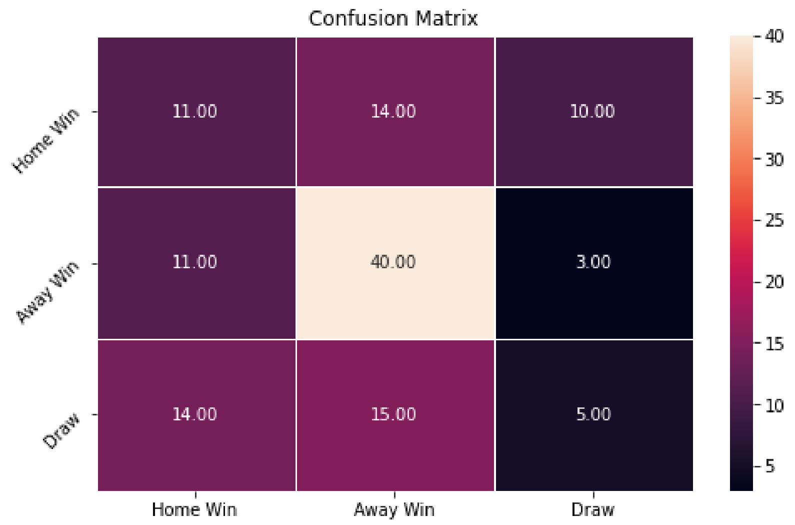


```
Accuracy Score : 0.617886

Classification Report:
               precision    recall  f1-score   support

    Home Win       0.44      0.23      0.30        35
    Away Win       0.72      0.81      0.77        54
        Draw       0.55      0.71      0.62        34

    accuracy                           0.62       123
   macro avg       0.57      0.58      0.56       123
weighted avg       0.59      0.62      0.59       123


AUC_ROC Score:  0.6940893866622098


Accuracy of Home Win = 0.6992
Accuracy of Away Win = 0.7805
Accuracy of Draw = 0.7561
```

K-Nearest Neighbors

The K-NN (K-Nearest Neighbor) algorithm is one of the simplest and most used classification algorithm. K-NN is a non-parametric, lazy learning algorithm. If we try to understand the concept of lazy, unlike eager learning, lazy learning does not have a training stage. It does not learn the training data but instead “memorizes” the training data set. When we want to make a guess, it looks for the closest neighbors in the entire data set. In the operation of the algorithm, a K value is determined. The meaning of this K value is the number of elements to look at. When a value comes, the distance between the value is calculated by taking the nearest K number of elements.

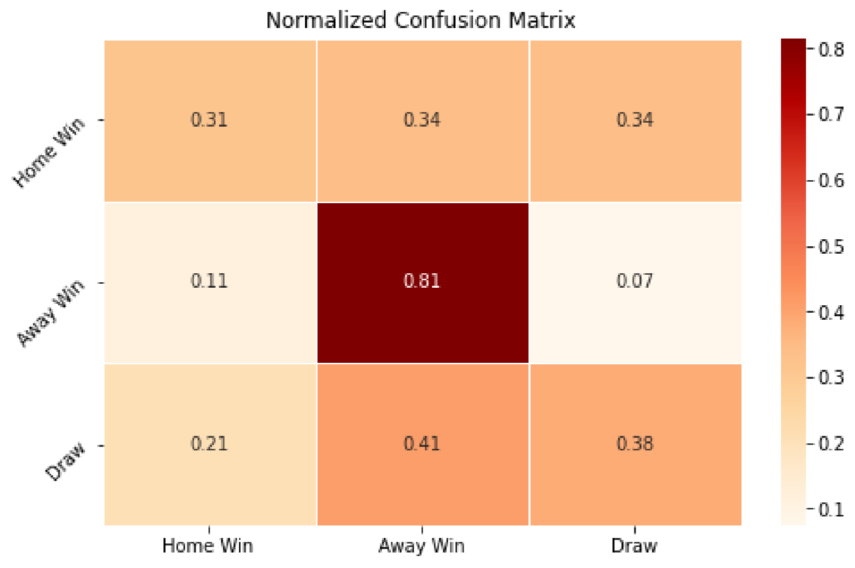


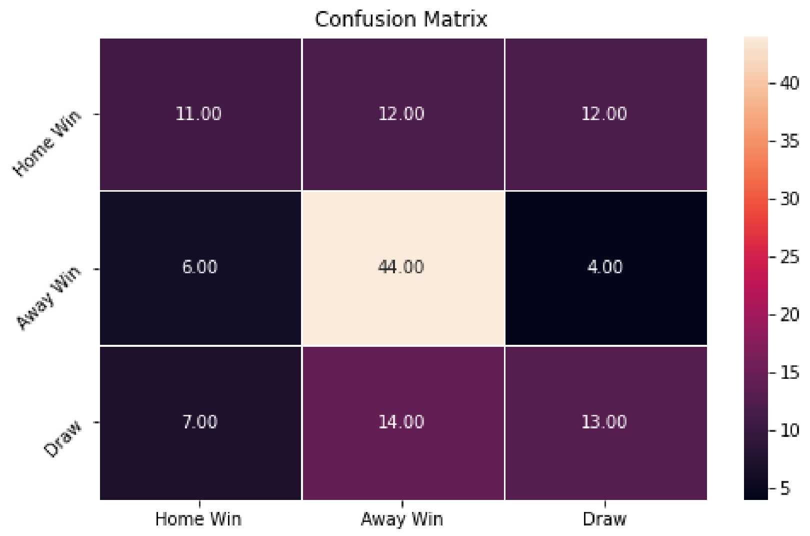


```
Accuracy Score : 0.455285

Classification Report:
              precision    recall  f1-score   support

    Home Win       0.31      0.31      0.31        35
    Away Win       0.58      0.74      0.65        54
        Draw       0.28      0.15      0.19        34

    accuracy                           0.46       123
   macro avg       0.39      0.40      0.38       123
weighted avg       0.42      0.46      0.43       123


AUC_ROC Score:  0.5586061831103595


Accuracy of Home Win = 0.6016
Accuracy of Away Win = 0.6504
Accuracy of Draw = 0.6585
```

Desicion Tree Classifier

Decision Trees try to maximize information gain by making choices that reduce the entropy value of the current situation. For this, it recalculates the error function in each question and selects the question / situation with the lowest error. In addition, Decision Trees are considered to be the closest to human thinking among machine learning models. Although the depth of some decision trees in people's minds is 1--2 as above, much more complex models are often constructed. For example, someone working in the human resources department might have a decision tree like the one below in their mind. If a person is in the x-y age range, graduates from a, b, c, d, e schools, has at least t years of experience and the average duration of work in previous jobs is more than p years, let's call for an interview.

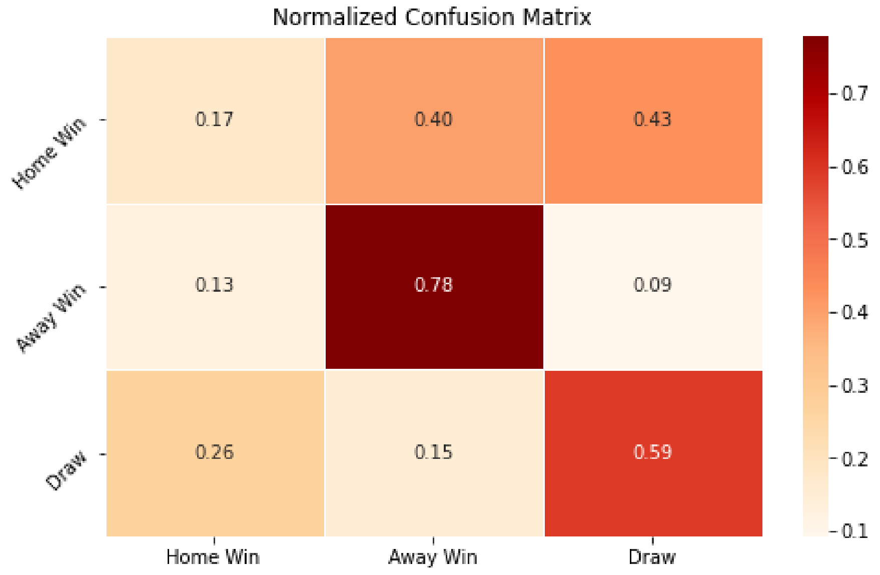


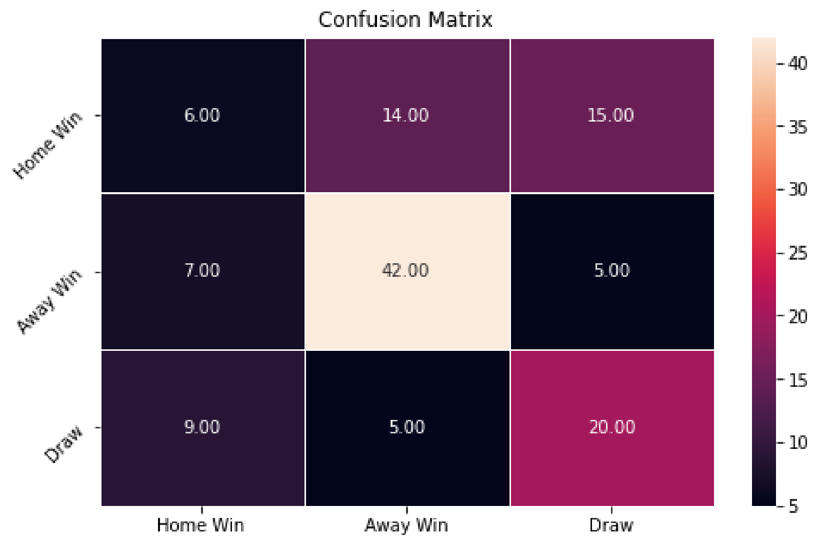


```
Accuracy Score : 0.552846

Classification Report:
              precision    recall  f1-score   support

    Home Win       0.46      0.31      0.37        35
    Away Win       0.63      0.81      0.71        54
        Draw       0.45      0.38      0.41        34

    accuracy                           0.55       123
   macro avg       0.51      0.50      0.50       123
weighted avg       0.53      0.55      0.53       123


AUC_ROC Score:  0.634523220407992


Accuracy of Home Win = 0.6992
Accuracy of Away Win = 0.7073
Accuracy of Draw = 0.6992
```

Support Vector Machine

Support Vector Machines are one of the supervised learning methods generally used in classification problems. Draws a line to separate points placed on a plane. This is aimed at keeping the line at the maximum distance for the points of both classes. It is suitable for complex but small to medium sized datasets.

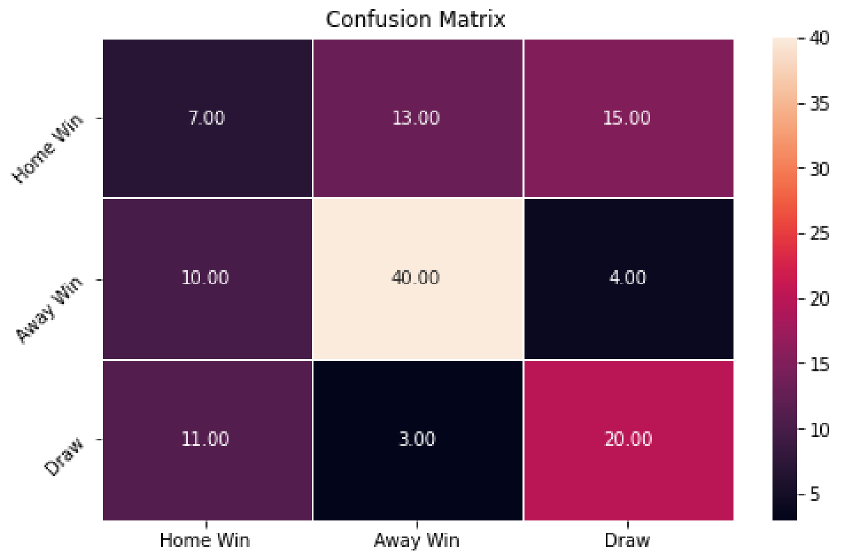


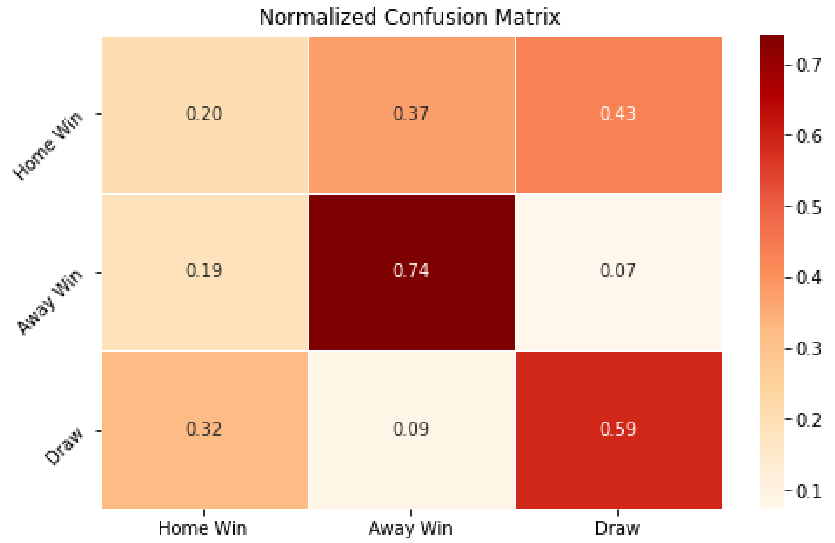


```
Accuracy Score : 0.552846

Classification Report:
               precision    recall  f1-score   support

    Home Win       0.27      0.17      0.21        35
    Away Win       0.69      0.78      0.73        54
        Draw       0.50      0.59      0.54        34

    accuracy                           0.55       123
   macro avg       0.49      0.51      0.49       123
weighted avg       0.52      0.55      0.53       123


AUC_ROC Score:  0.6425903402569398


Accuracy of Home Win = 0.6341
Accuracy of Away Win = 0.7480
Accuracy of Draw = 0.7236
```

Gaussian Naive Bayes

The Naive Bayes classifier is a probabilistic machine learning model used for the classification task. The point of the classifier is based on Bayes' theorem. Using Bayes' theorem, we can find the probability that A occurs, given the occurrence of B. Here B is the proof and A is the hypothesis. The assumption made here is that the predictors / characteristics are independent. This means that the presence of one particular feature does not affect another. That's why it is called pure. When the estimators take a continuous value and are not discrete, we assume that these values are sampled from a gaussian distribution.





```
Accuracy Score : 0.544715

Classification Report:
               precision    recall  f1-score   support

    Home Win       0.25      0.20      0.22        35
    Away Win       0.71      0.74      0.73        54
        Draw       0.51      0.59      0.55        34

    accuracy                           0.54       123
   macro avg       0.49      0.51      0.50       123
weighted avg       0.53      0.54      0.53       123


AUC_ROC Score:  0.640828744530599


Accuracy of Home Win = 0.6016
Accuracy of Away Win = 0.7561
Accuracy of Draw = 0.7317
```

Conclusion

In this study, we compared cardiotocogram data using data science techniques best suited for classification.

We analyzed our data before applying data science techniques. As a result of our analysis, we realized that our data is very clean. When we got to the data cleaning stage, we saw that there was no duplicate, null and NaN data. We checked the relevance and importance of the features to our target feature. We removed low correlation and redundant features from our dataset.

We used 5 different classification models to build our model to predict match outcome.

1- Random Forest Classifier

2- K-Nearest Neighbors

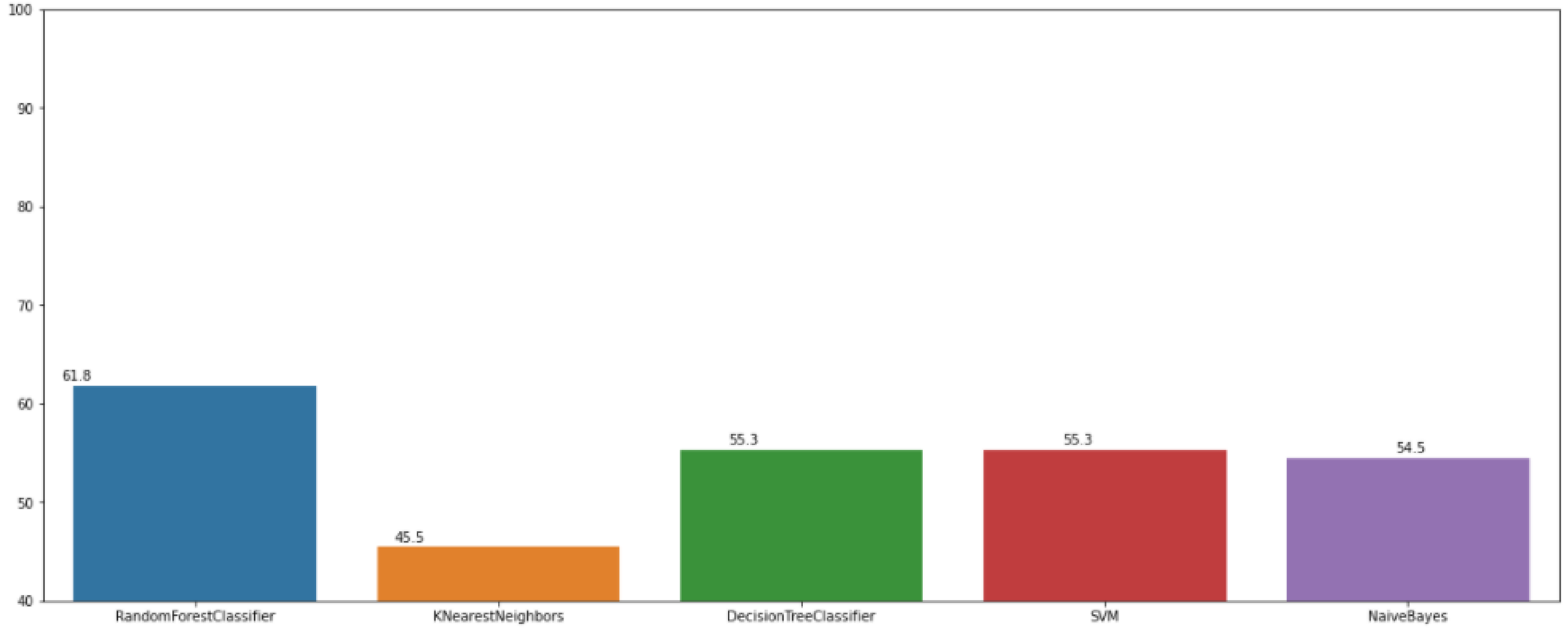
3- Decision Tree Classifier

4- Support Vector Machine

5- Gaussian Naive Bayes

These models include both linear and Tree-based models. We saw that parameter setting is very important while creating the model, and we determined the most suitable parameters for the data we have by doing Grid Search. With this method, we achieved a 5% increase in model accuracy. We tried many ways to improve the accuracy of the model, but due to low match data, we could not try deep learning methods. For this reason, we achieved the highest accuracy of 62% with Random Forest, which is included in classical machine learning methods.

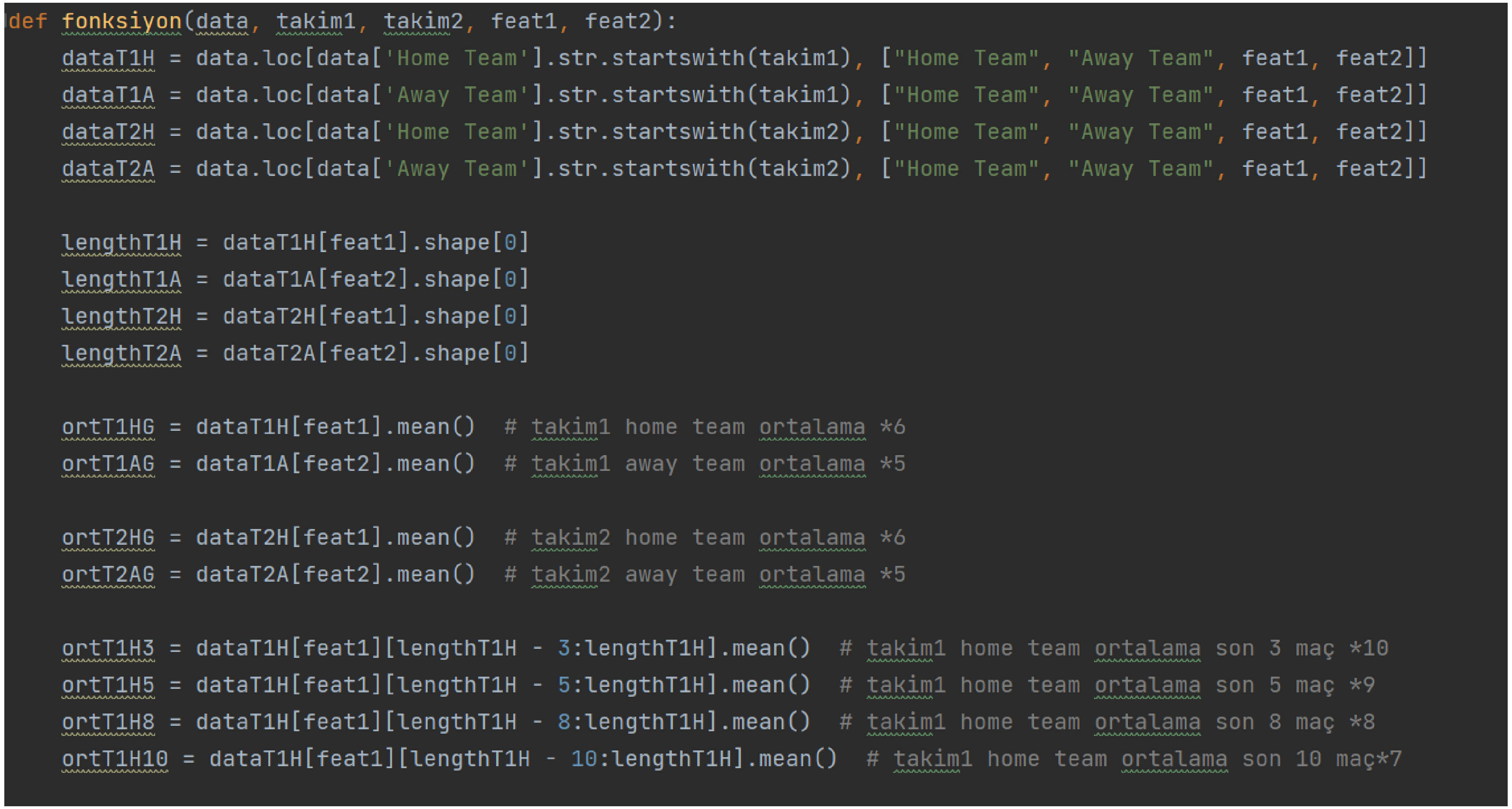
Although 62% accuracy does not seem sufficient, we aim to increase this rate to 75-80% with deep learning methods as the data increases.


100
90
80
70
60
50
40
61.8
45.5
55.3
55.3
54.5
RandomForestClassifier
KNearestNeighbors
DecisionTreeClassifier
SVM
NaiveBayes

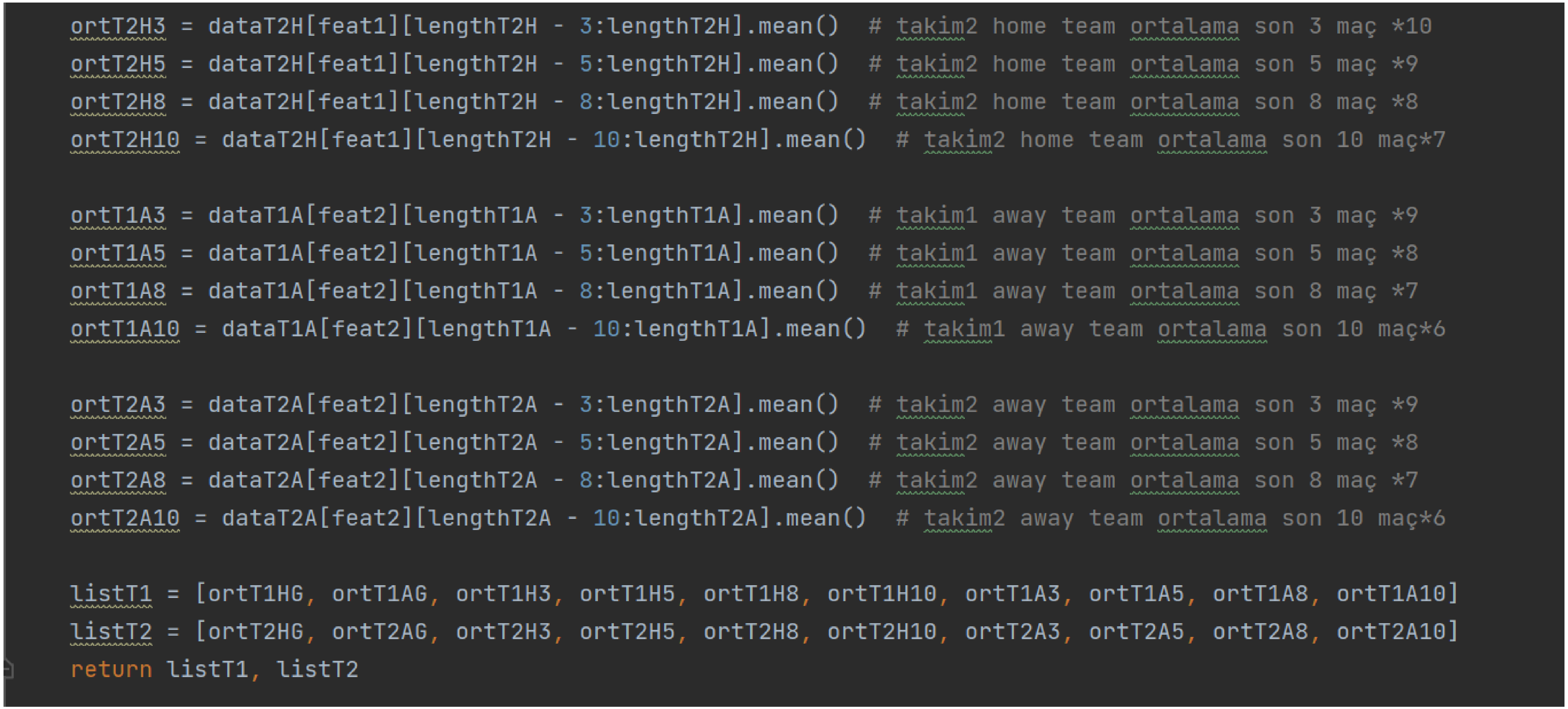
Feature Prediction

In order to predict a future match, we also need to predict the features we will use in our main model. To solve this problem, we took the average of the statistics of the previous matches played between the teams and put them into the models we created with real values, including the last 3,5,7,10 matches played.

Finding the average of matches between two teams

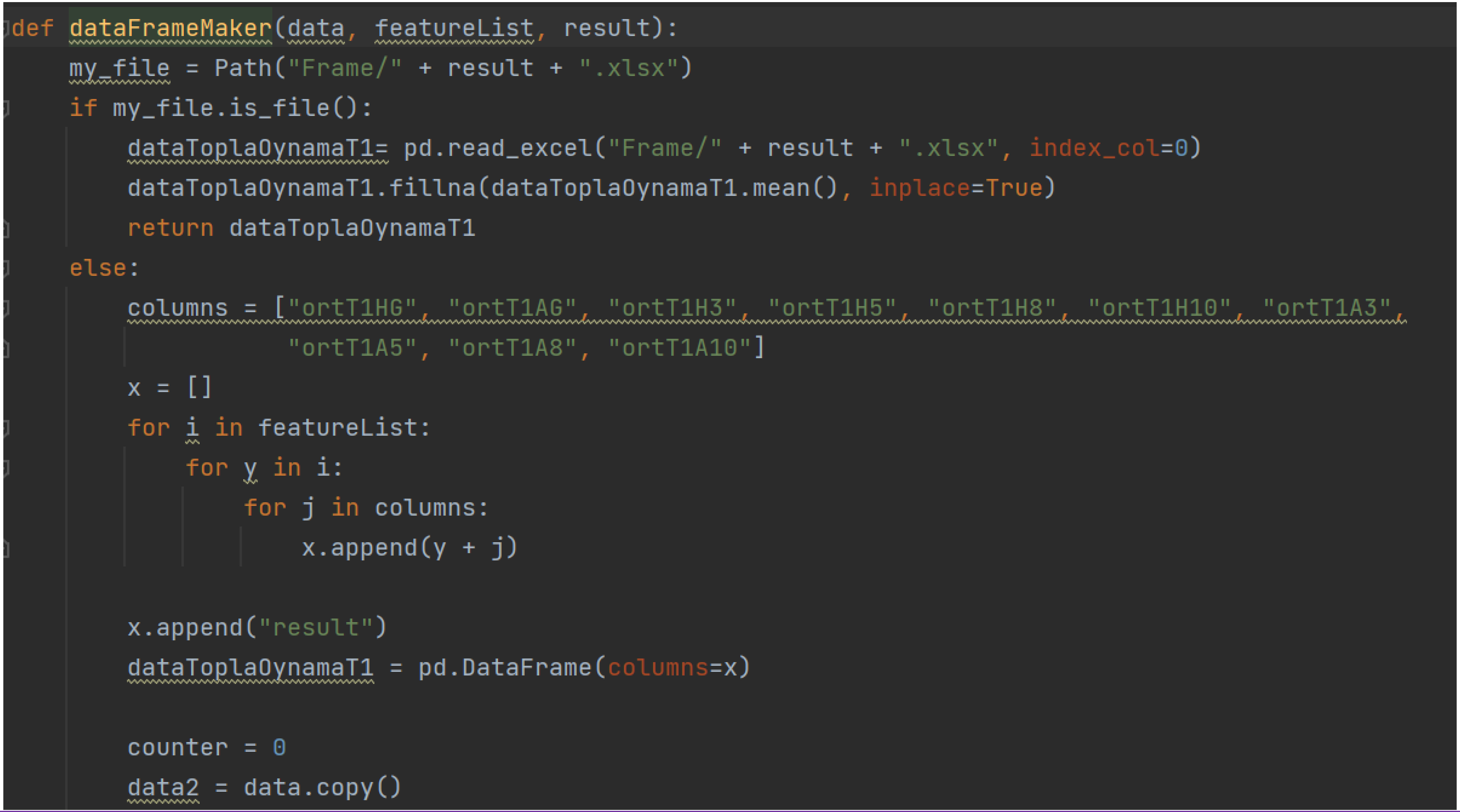


```python
def fonksiyon(data, takim1, takim2, feat1, feat2):
    dataT1H = data.loc[data['Home Team'].str.startswith(takim1), ["Home Team", "Away Team", feat1, feat2]]
    dataT1A = data.loc[data['Away Team'].str.startswith(takim1), ["Home Team", "Away Team", feat1, feat2]]
    dataT2H = data.loc[data['Home Team'].str.startswith(takim2), ["Home Team", "Away Team", feat1, feat2]]
    dataT2A = data.loc[data['Away Team'].str.startswith(takim2), ["Home Team", "Away Team", feat1, feat2]]

    lengthT1H = dataT1H[feat1].shape[0]
    lengthT1A = dataT1A[feat2].shape[0]
    lengthT2H = dataT2H[feat1].shape[0]
    lengthT2A = dataT2A[feat2].shape[0]

    ortT1HG = dataT1H[feat1].mean()  # takim1 home team ortalama *6
    ortT1AG = dataT1A[feat2].mean()  # takim1 away team ortalama *5

    ortT2HG = dataT2H[feat1].mean()  # takim2 home team ortalama *6
    ortT2AG = dataT2A[feat2].mean()  # takim2 away team ortalama *5

    ortT1H3 = dataT1H[feat1][lengthT1H - 3:lengthT1H].mean()  # takim1 home team ortalama son 3 maç *10
    ortT1H5 = dataT1H[feat1][lengthT1H - 5:lengthT1H].mean()  # takim1 home team ortalama son 5 maç *9
    ortT1H8 = dataT1H[feat1][lengthT1H - 8:lengthT1H].mean()  # takim1 home team ortalama son 8 maç *8
    ortT1H10 = dataT1H[feat1][lengthT1H - 10:lengthT1H].mean()  # takim1 home team ortalama son 10 maç*7
```

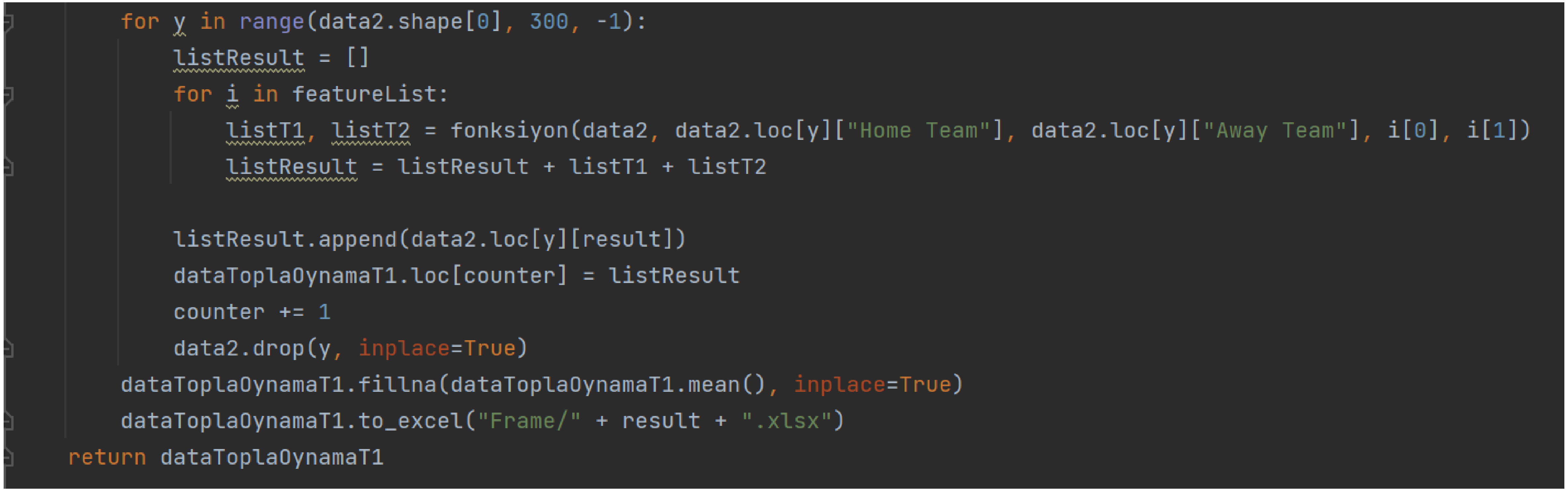


```
    ortT2H3 = dataT2H[feat1][lengthT2H - 3:lengthT2H].mean()  # takim2 home team ortalama son 3 maç *10
    ortT2H5 = dataT2H[feat1][lengthT2H - 5:lengthT2H].mean()  # takim2 home team ortalama son 5 maç *9
    ortT2H8 = dataT2H[feat1][lengthT2H - 8:lengthT2H].mean()  # takim2 home team ortalama son 8 maç *8
    ortT2H10 = dataT2H[feat1][lengthT2H - 10:lengthT2H].mean()  # takim2 home team ortalama son 10 maç*7

    ortT1A3 = dataT1A[feat2][lengthT1A - 3:lengthT1A].mean()  # takim1 away team ortalama son 3 maç *9
    ortT1A5 = dataT1A[feat2][lengthT1A - 5:lengthT1A].mean()  # takim1 away team ortalama son 5 maç *8
    ortT1A8 = dataT1A[feat2][lengthT1A - 8:lengthT1A].mean()  # takim1 away team ortalama son 8 maç *7
    ortT1A10 = dataT1A[feat2][lengthT1A - 10:lengthT1A].mean()  # takim1 away team ortalama son 10 maç*6

    ortT2A3 = dataT2A[feat2][lengthT2A - 3:lengthT2A].mean()  # takim2 away team ortalama son 3 maç *9
    ortT2A5 = dataT2A[feat2][lengthT2A - 5:lengthT2A].mean()  # takim2 away team ortalama son 5 maç *8
    ortT2A8 = dataT2A[feat2][lengthT2A - 8:lengthT2A].mean()  # takim2 away team ortalama son 8 maç *7
    ortT2A10 = dataT2A[feat2][lengthT2A - 10:lengthT2A].mean()  # takim2 away team ortalama son 10 maç*6

    listT1 = [ortT1HG, ortT1AG, ortT1H3, ortT1H5, ortT1H8, ortT1H10, ortT1A3, ortT1A5, ortT1A8, ortT1A10]
    listT2 = [ortT2HG, ortT2AG, ortT2H3, ortT2H5, ortT2H8, ortT2H10, ortT2A3, ortT2A5, ortT2A8, ortT2A10]
    return listT1, listT2
```

In the DataFrameMaker function, we call the function that finds the average for the 300 records of the data set we have for each of the 30 features we have specified for the model. Then it saves these data frames as an excel file. Due to our performance concerns in the project, every time this function is called, it also checks whether the feature name it takes as a parameter is in the project directory of the excel file.

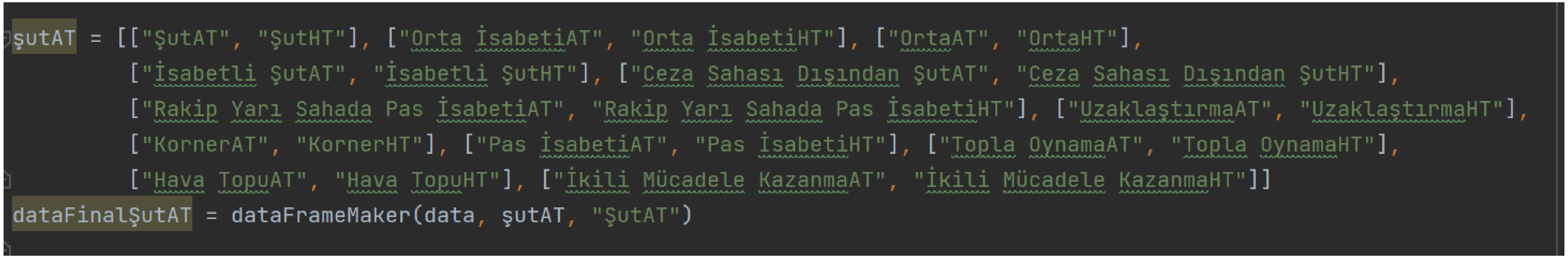


```
def dataFrameMaker(data, featureList, result):
    my_file = Path("Frame/" + result + ".xlsx")
    if my_file.is_file():
        dataToplaOynamaT1= pd.read_excel("Frame/" + result + ".xlsx", index_col=0)
        dataToplaOynamaT1.fillna(dataToplaOynamaT1.mean(), inplace=True)
        return dataToplaOynamaT1
    else:
        columns = ["ortT1HG", "ortT1AG", "ortT1H3", "ortT1H5", "ortT1H8", "ortT1H10", "ortT1A3",
                   "ortT1A5", "ortT1A8", "ortT1A10"]
        x = []
        for i in featureList:
            for y in i:
                for j in columns:
                    x.append(y + j)

        x.append("result")
        dataToplaOynamaT1 = pd.DataFrame(columns=x)

        counter = 0
        data2 = data.copy()
```



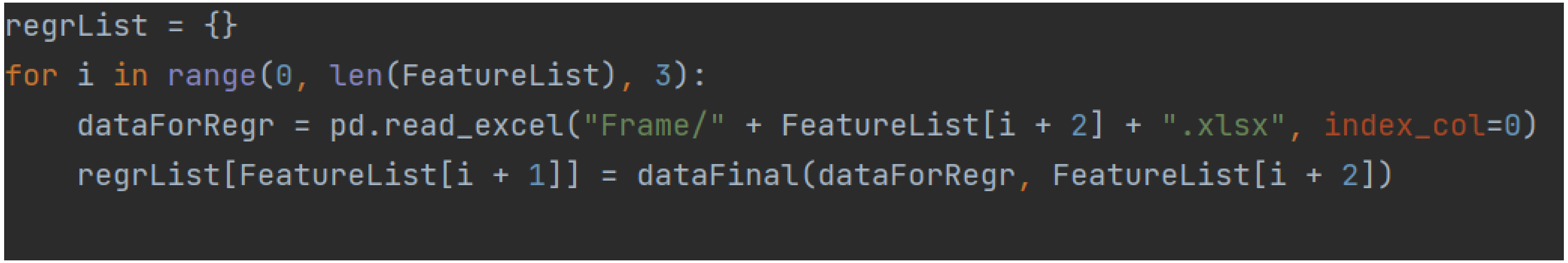
```
        for y in range(data2.shape[0], 300, -1):
            listResult = []
            for i in featureList:
                listT1, listT2 = fonksiyon(data2, data2.loc[y]["Home Team"], data2.loc[y]["Away Team"], i[0], i[1])
                listResult = listResult + listT1 + listT2

            listResult.append(data2.loc[y][result])
            dataToplaOynamaT1.loc[counter] = listResult
            counter += 1
            data2.drop(y, inplace=True)
        dataToplaOynamaT1.fillna(dataToplaOynamaT1.mean(), inplace=True)
        dataToplaOynamaT1.to_excel("Frame/" + result + ".xlsx")
    return dataToplaOynamaT1
```

Each list sent to the DataFrameMaker function holds a list of other features that will be used to predict the feature.

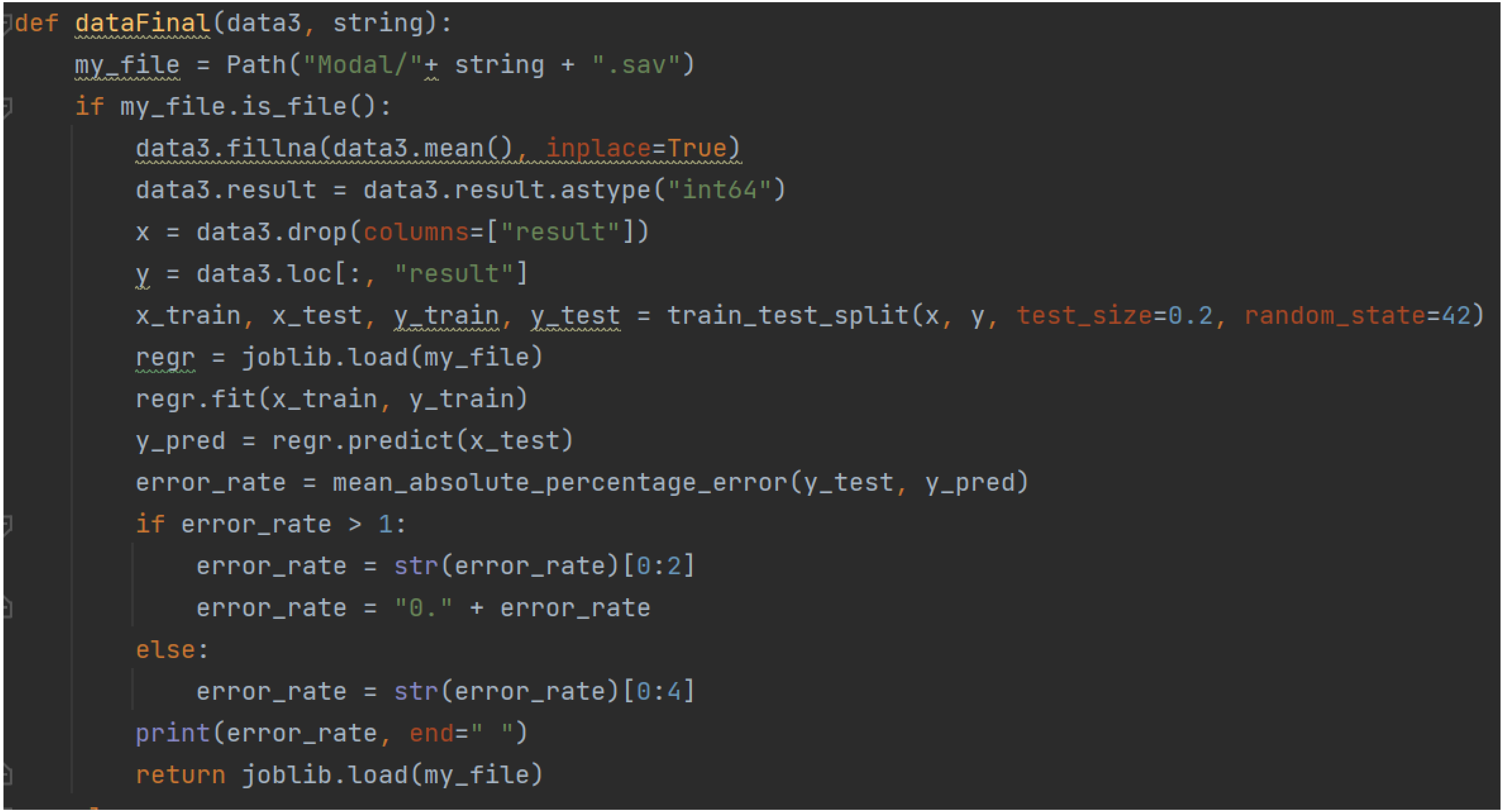


```
şutAT = [["ŞutAT", "ŞutHT"], ["Orta İsabetiAT", "Orta İsabetiHT"], ["OrtaAT", "OrtaHT"],
         ["İsabetli ŞutAT", "İsabetli ŞutHT"], ["Ceza Sahası Dışından ŞutAT", "Ceza Sahası Dışından ŞutHT"],
         ["Rakip Yarı Sahada Pas İsabetiAT", "Rakip Yarı Sahada Pas İsabetiHT"], ["UzaklaştırmaAT", "UzaklaştırmaHT"],
         ["KornerAT", "KornerHT"], ["Pas İsabetiAT", "Pas İsabetiHT"], ["Topla OynamaAT", "Topla OynamaHT"],
         ["Hava TopuAT", "Hava TopuHT"], ["İkili Mücadele KazanmaAT", "İkili Mücadele KazanmaHT"]]
dataFinalŞutAT = dataFrameMaker(data, şutAT, "ŞutAT")
```

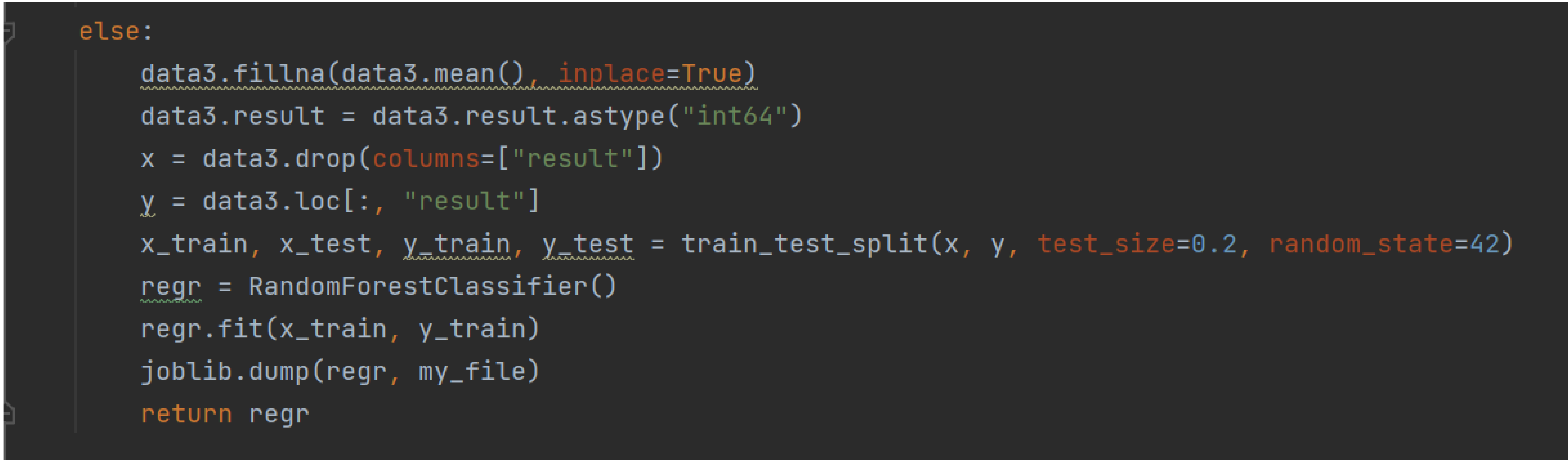
In the DataFinalFunction, the models of the features we will use in match statistics are trained. If a model has been trained before, it is not retrained. These trained models are kept in a list. The name of this list is regrList.



```
regrList = {}
for i in range(0, len(FeatureList), 3):
    dataForRegr = pd.read_excel("Frame/" + FeatureList[i + 2] + ".xlsx", index_col=0)
    regrList[FeatureList[i + 1]] = dataFinal(dataForRegr, FeatureList[i + 2])
```

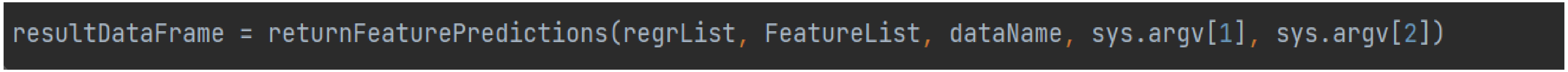


```
def dataFinal(data3, string):
    my_file = Path("Modal/"+ string + ".sav")
    if my_file.is_file():
        data3.fillna(data3.mean(), inplace=True)
        data3.result = data3.result.astype("int64")
        x = data3.drop(columns=["result"])
        y = data3.loc[:, "result"]
        x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
        regr = joblib.load(my_file)
        regr.fit(x_train, y_train)
        y_pred = regr.predict(x_test)
        error_rate = mean_absolute_percentage_error(y_test, y_pred)
        if error_rate > 1:
            error_rate = str(error_rate)[0:2]
            error_rate = "0." + error_rate
        else:
            error_rate = str(error_rate)[0:4]
        print(error_rate, end=" ")
        return joblib.load(my_file)
```

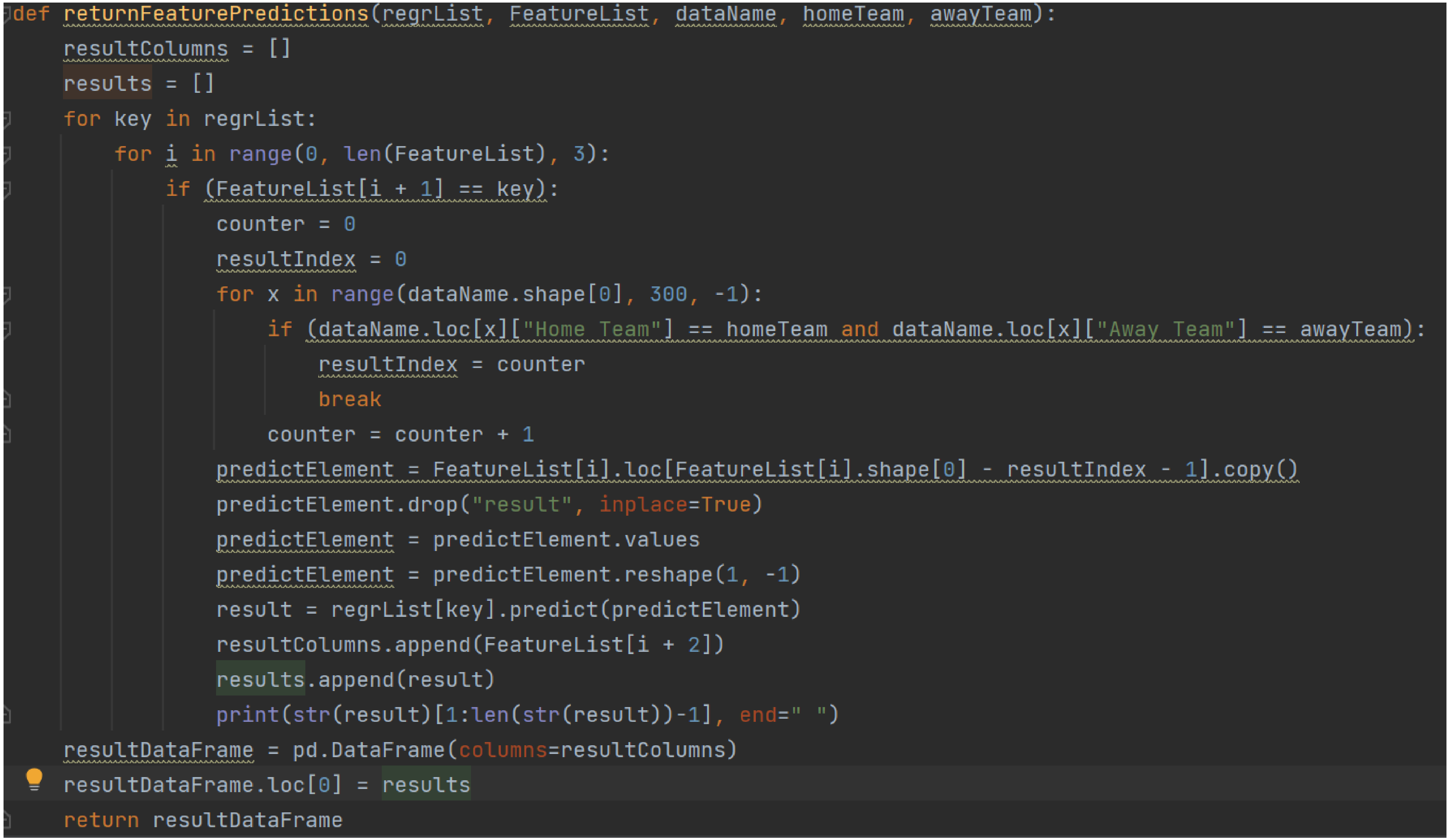


```
else:
    data3.fillna(data3.mean(), inplace=True)
    data3.result = data3.result.astype("int64")
    x = data3.drop(columns=["result"])
    y = data3.loc[:, "result"]
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
    regr = RandomForestClassifier()
    regr.fit(x_train, y_train)
    joblib.dump(regr, my_file)
    return regr
```

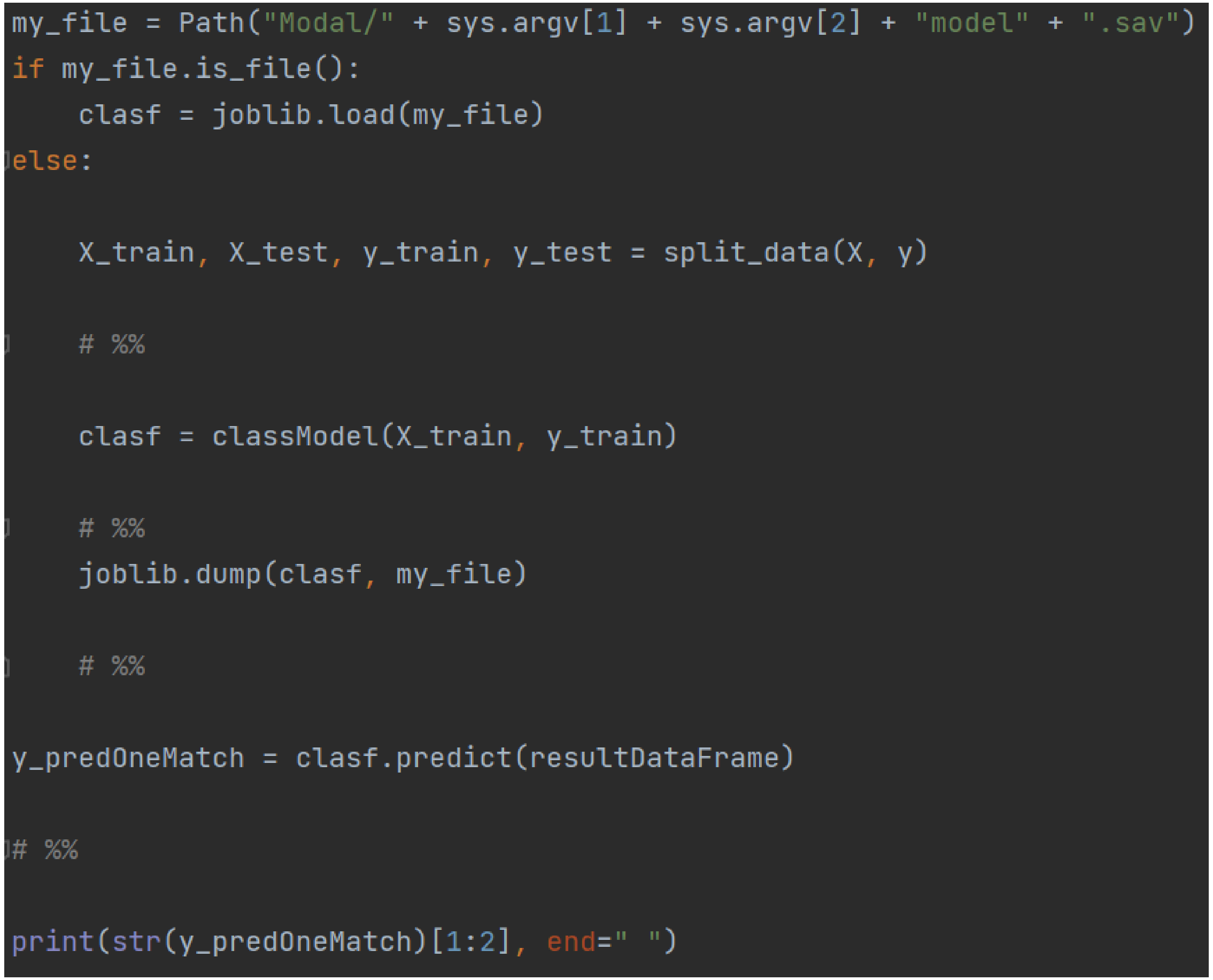
Afterwards, in the match to be played between the two teams, the statistics of each feature of the last 3,5,8,10 matches are found starting from the last match, and the prediction results of the features are predicted to the previously trained features in the regrList with these values. By predicting these obtained results in our main model, we get the predicted result of the match. These predicted features and their accuracy rates and the result of the match are transferred to our backend application via console.



```
resultDataFrame = returnFeaturePredictions(regrList, FeatureList, dataName, sys.argv[1], sys.argv[2])
```



```python
def returnFeaturePredictions(regrList, FeatureList, dataName, homeTeam, awayTeam):
    resultColumns = []
    results = []
    for key in regrList:
        for i in range(0, len(FeatureList), 3):
            if (FeatureList[i + 1] == key):
                counter = 0
                resultIndex = 0
                for x in range(dataName.shape[0], 300, -1):
                    if (dataName.loc[x]["Home Team"] == homeTeam and dataName.loc[x]["Away Team"] == awayTeam):
                        resultIndex = counter
                        break
                    counter = counter + 1
                predictElement = FeatureList[i].loc[FeatureList[i].shape[0] - resultIndex - 1].copy()
                predictElement.drop("result", inplace=True)
                predictElement = predictElement.values
                predictElement = predictElement.reshape(1, -1)
                result = regrList[key].predict(predictElement)
                resultColumns.append(FeatureList[i + 2])
                results.append(result)
                print(str(result)[1:len(str(result))-1], end=" ")
    resultDataFrame = pd.DataFrame(columns=resultColumns)
    resultDataFrame.loc[0] = results
    return resultDataFrame
```



```python
my_file = Path("Modal/" + sys.argv[1] + sys.argv[2] + "model" + ".sav")
if my_file.is_file():
    clasf = joblib.load(my_file)
else:

    X_train, X_test, y_train, y_test = split_data(X, y)

    # %%

    clasf = classModel(X_train, y_train)

    # %%
    joblib.dump(clasf, my_file)

    # %%

y_predOneMatch = clasf.predict(resultDataFrame)

# %%

print(str(y_predOneMatch)[1:2], end=" ")
```