Ayça Avcı - 53961

Barış Baturay - 54052

QMBU450

Final Project Report

Exploratory Goal

In this project, we implemented a machine learning algorithm in Python using the scikit-learn library for a real-life regression problem from the finance industry. Our exploratory goal is to predict the number of cash withdrawals for 10 days (between June 1, 2019, and June 10, 2019) from 47 different ATMs of a private bank in Turkey using the information given about each ATM and the withdrawal date. To achieve this goal, we used the Random Forest Regressor algorithm from the scikit-learn Python library.

Data Set

We have two input data files, namely, training_data.csv and test_data.csv. The training set contains 42,958 labeled data instances (47 ATMs x 457 days x 2 transaction types), where each training data instance has 7 columns. The IDENTITY column gives us the unique identifier assigned to each ATM. REGION column shows the geographical region of each ATM. DAY, MONTH, and YEAR columns give the transaction date. TRX_TYPE column shows the transaction type whether the card is present or not (1: card present, 2: card not present). TRX_COUNT is the number of cash withdrawals performed on the specified date. We used the trained machine learning algorithm to perform predictions for the test data set, which contains 940 data instances (47 ATMs x 10 days x 2 transaction types). Since we do not have the given numbers of cash withdrawals for test instances, we split the training set to two, as training data set and test data set respectively. The predictive quality of our solution is evaluated in terms of its MAE (mean absolute error) and RMSE (root mean squared error) values on this test set.

Data Preprocessing

As a first step, we read both test and training data using read.csv() function from the pandas library. We assigned them to train_X and test_X variables. We retrieved the TRX_COUNT column and assigned it as train_Y variable, after that, we dropped that column from the train_X variable since we are going to make TRX_COUNT predictions. We use train_X and train_Y variables in train_test_split function. Since the IDENTITY column consists of 9-digit values and corresponds to 47 different ATMs, we made a one-hot encoding on the IDENTITY column using pandas get_dummies() function and dropped this column from the data set afterwards. One-hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. After that, we inserted a new feature called WEEKEND to the train_X to capture the change in cash withdrawals on weekends and weekdays. We split train_X and train_Y as a test and train data using train_test_split() function from scikit-learn. We gave 0.2 to the test size parameter (a small part of the actual train data) to train my model on more data points.

Random Forest Regressor

Random Forest is based on the bagging algorithm and uses the Ensemble Learning technique. It improves on bagging because it decorrelates the trees with the introduction of splitting on a random subset of features on each split. It creates as many trees on the subset of the data and combines the output of all the trees. In this way, it reduces overfitting problems in decision trees and also reduces the variance and therefore improves the accuracy. It can handle binary features, categorical features, and numerical features. There is very little pre-processing that needs to be done. The data does not need to be rescaled or transformed. Random forests are great with high dimensional data, like ours, since we are working with subsets of data.

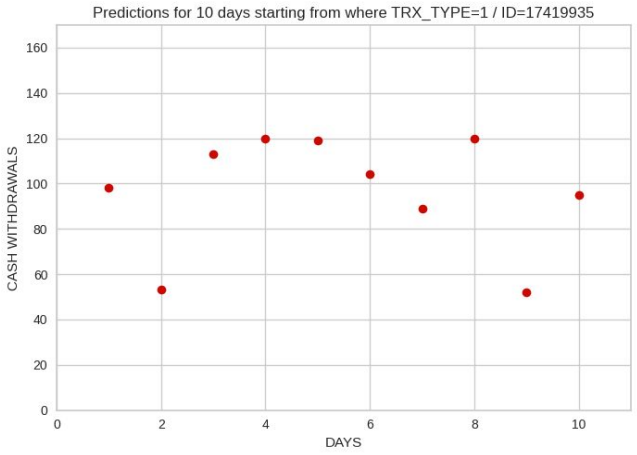
We run the Exhaustive Grid Search, GridSearchCV from the scikit-learn library, using KUACC (Koç University Advanced Computing Center) to find best parameters for our model. GridSearchCV gave the best values for parameters as follows:

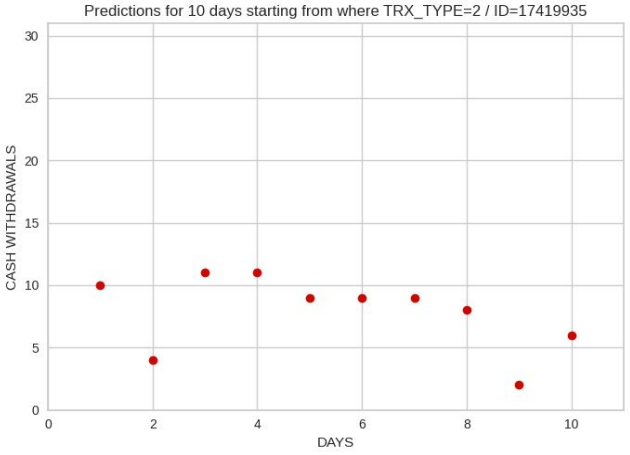
n_estimators=200, max_depth=30, min_samples_split=2, min_samples_leaf=5

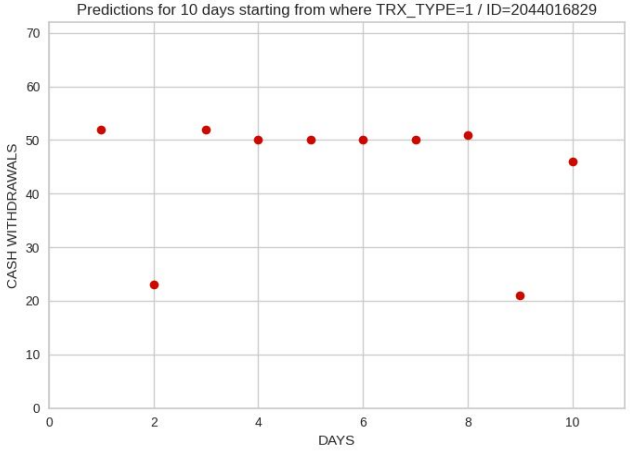
Each of these parameters mean the number of trees in the forest, the maximum depth of the tree, the minimum number of samples required to split an internal node, and the minimum number of samples required to be at a leaf node respectively. We trained our model according to these parameter values.

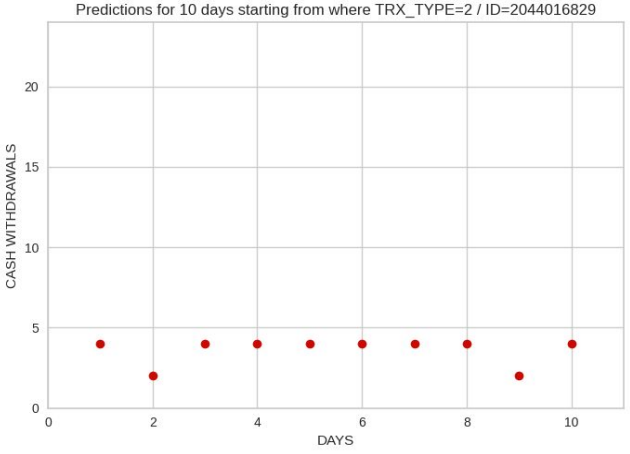
Findings and Results

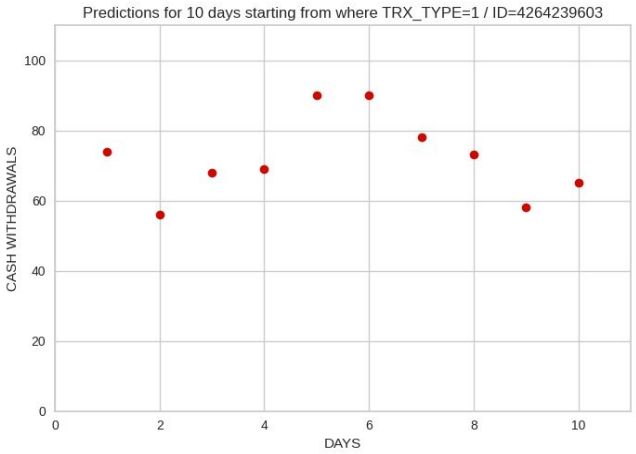
After that, we fit our training data to our model and we predicted y_predict values for test data. We calculated root mean squared error and mean absolute error, and found 14.1898 and 8.2051 respectively. As a final step, we made predictions on test_data.csv and wrote the results in a CSV file. Since we do not have the correct number of cash withdrawals for these 10 days, we could not do a comparison between predicted results and real results. You can see a couple of graphs for our predictions as below. TRX_TYPE means whether a card is used for withdrawal or not (card present: 1, card is not present: 2) and ID represents the ID of the ATM.

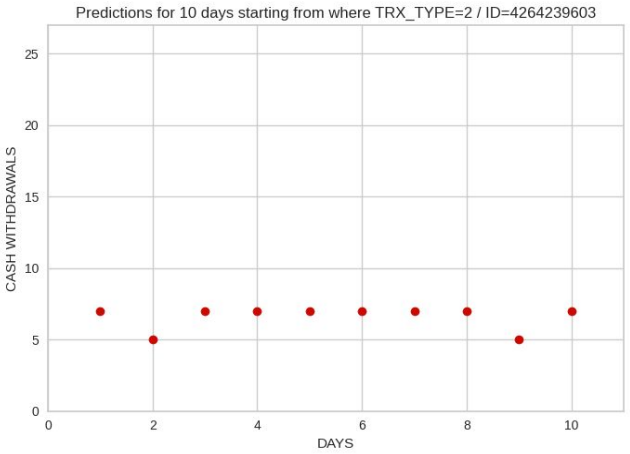


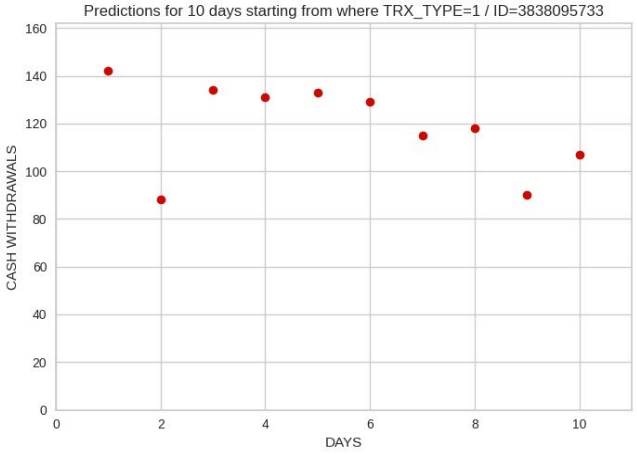


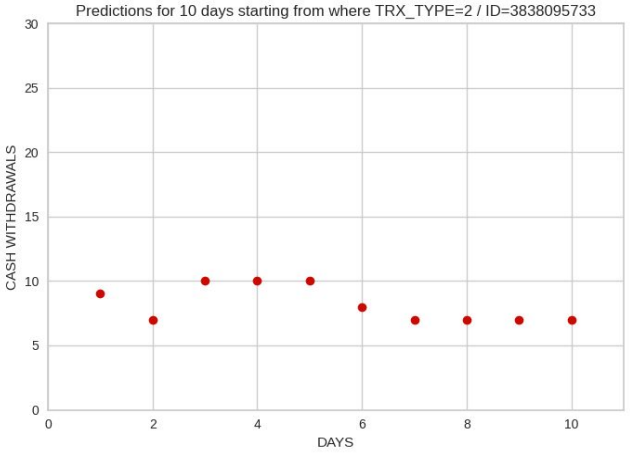


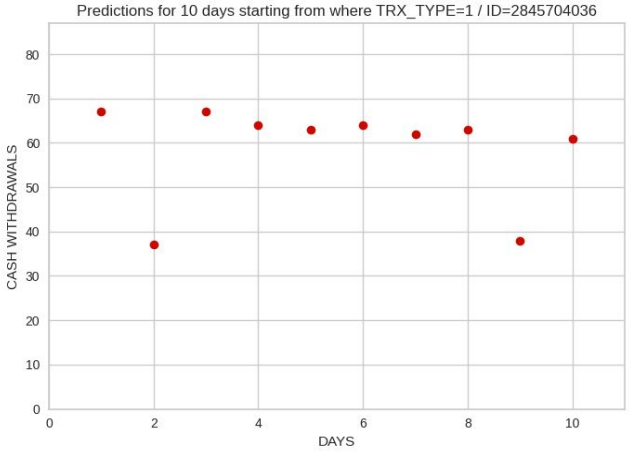


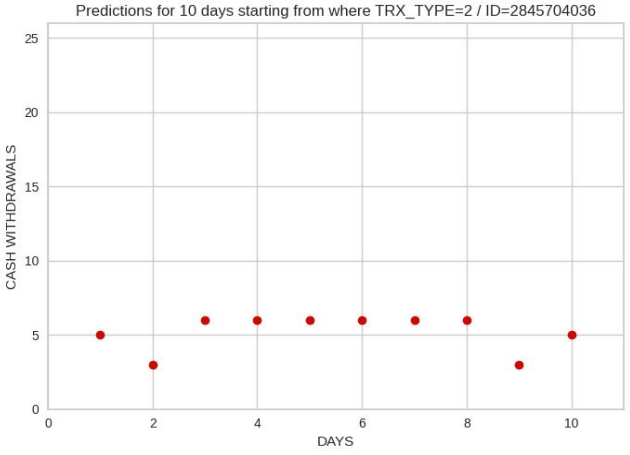

Predictions for 10 days starting from where TRX_TYPE=1 / ID=4264239603
CASH WITHDRAWALS
DAYS


Predictions for 10 days starting from where TRX_TYPE=2 / ID=4264239603
CASH WITHDRAWALS
DAYS


Predictions for 10 days starting from where TRX_TYPE=1 / ID=3838095733
CASH WITHDRAWALS
DAYS


Predictions for 10 days starting from where TRX_TYPE=2 / ID=3838095733
CASH WITHDRAWALS
DAYS


Predictions for 10 days starting from where TRX_TYPE=1 / ID=2845704036
CASH WITHDRAWALS
DAYS


Predictions for 10 days starting from where TRX_TYPE=2 / ID=2845704036
CASH WITHDRAWALS
DAYS