ML Hackathon

K V V Deepesh(IMT2019508)

K Yashovardhan Reddy(IMT2019097)

Regression Problem

Use Regression model to predict the values in the test dataset with 2 features, from the train dataset.

Preprocessing

- There were no NaN or '?' values in the train and test dataset. Thus not much pre-processing was required.

- Used StandardScaler and MinMaxScaler of scikit-learn to normalise the data.

Model Description

- Initially a test train split was done, where 20% of the data after shuffling was reserved for testing.

- After plotting the graph of feature vs target variable(Figure 1 and 2), it was concluded that the features were not linearly related to the target variable.

- PolynomialFeatures from scikit learn has been used to make new columns, which are the polynomial combination of existing features, then linear regression has been done on those features.

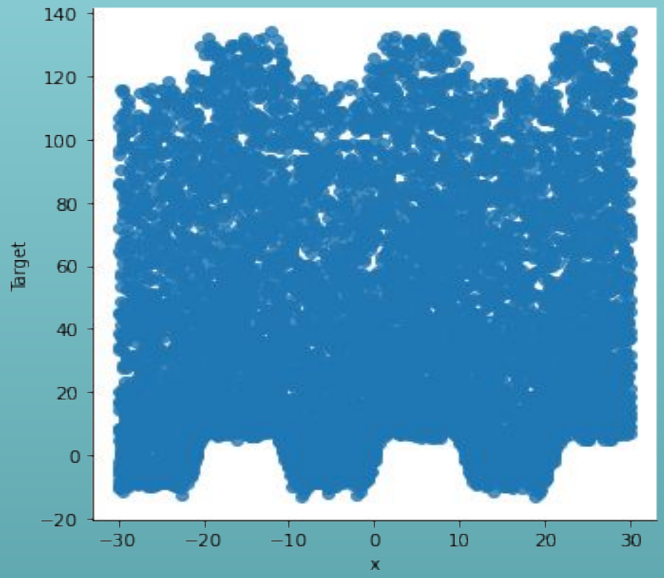

Fig.1 Feature ‘x’ vs target variable

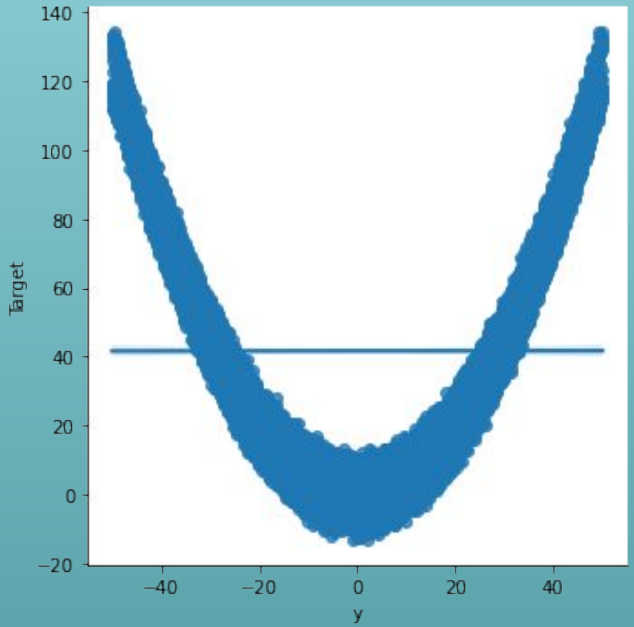

Fig.2 Feature ‘y’ vs target variable.

Hyperparameters and Kaggle score

- The degree of the polynomial to fit was the only hyperparameter that needed to be changed.
- Experimenting with various degrees and inferring from the plots we thought degree of 10 would fit the data well and anything more would cause overfitting of the model.
- Degree 10 and a MinMaxScaler or StandardScaler gave us the best kaggle score.

Classification Problem

Use Classification model to predict the outcome in the test dataset from the train dataset.

Preprocessing

- There were almost 300 features in the dataset.
- Combined both test and train dataset into a single dataset to make them easier to work with.
- After checking for NaN values, dropped two columns which more than 30% of their missing.
- For the remaining columns, replaced the NaN values with mode of the column i.e. most common element.

- The column ‘Field7’ had numbers as data but was of the type ‘object’. So the values in the column were all converted to the type ‘int’.

- For the remaining categorical columns, columns which had only two unique variables ‘Y’ and ‘N’ were encoded such that Y : 1 and N : 0.

- All the remaining categorical columns were dropped.

Correlation

- Used the .corr() function in pandas to delete highly correlated columns.
- Columns with correlation greater than 0.95 were dropped.

Models used:

- Used StandardScaler of scikit-learn to normalize the data. Initially tried to predict the outcome using Naive Bayes Model to predict the outcome.

- After using Naives Bayes algorithm, used Logistic Regression model of scikit learn to predict the outcome which gave much better results.

- After using Logistic Regression, tried to use SVM classifier which did not give better results than Logistic Regression.

- Also tried one-hot encoding the categorical columns which were dropped previously, this did not produce any better results and increased the number of columns to 400.

Hyperparameters and Kaggle score

- Maximum number of iterations, class weight, solver and C are the hyper-parameters in the Logistic Regression model.

- Experimenting with various hyper-parameters and models led to the conclusion that the Logistic Regression model led to the highest Kaggle score.

Challenges faced while creating model

- While dealing with NaN values, as the number of NaN was very less for some columns we deleted the rows. This created as some rows in test dataset were also deleted. To deal with this problem, we replace the missing values with the mode instead of deleting the rows.

- Reducing the number of columns was a big issue, especially after using one-hot encoding, the number of columns increased to 600. To deal with we decided not to use one hot encoding and instead used label encoding for some columns and dropped other columns.

- To reduce the number of columns further we used correlation matrix and dropped columns with correlation greater than 0.97 .

THE END