CE802 Machine Learning and Data Mining

YASWANTH MARAM

Registration Number: 2003287

University of Essex

Mathematical Sciences Department

December 31, 2020

This report contains 1159 words

**4. Report on the Investigation**

**Part – 2**

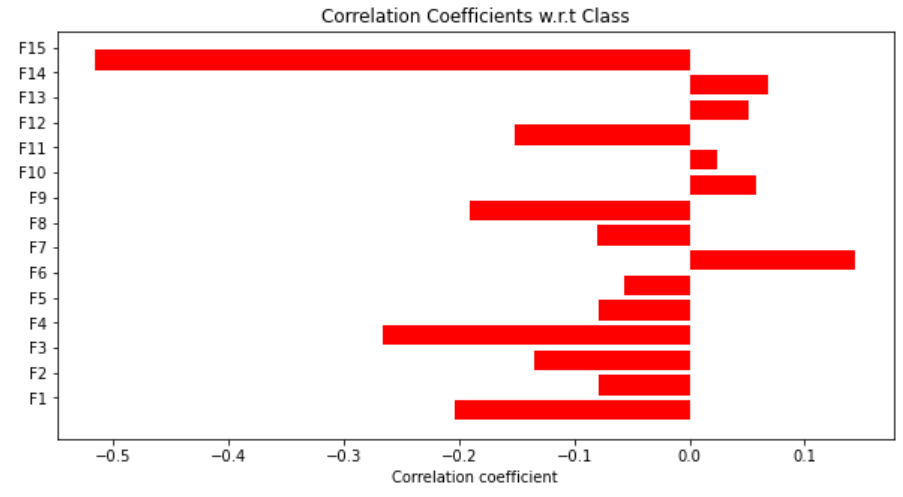
# Data:

The Train dataset consists of historical data made of 1500 examples with 15 features starting from "F1" to "F15" and one output label "Class" representing whether the insured filed a claim or not. On the other hand, test dataset comprises of feature from "F1" to "F15" with missing output label class to be replaced with the output predictions.

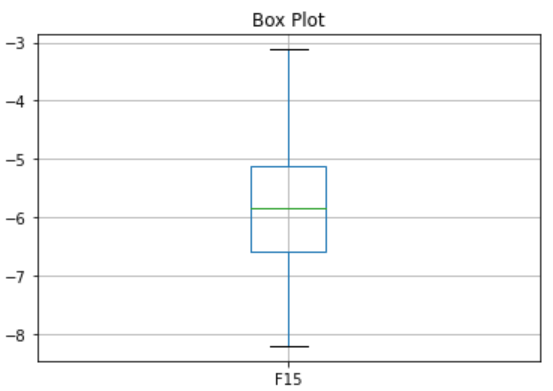
Any Machine Learning Project has a few steps that needs to be done for getting the output like Data Pre-processing, Feature Engineering, Model Building, Hyper-parameter Tuning and finally Evaluation of Model.

# Pre-processing & Feature Engineering:

1. It is observed that the data contains missing values in feature F15. If you see below graph you can clearly see that there is a ***‘high correlation’*** between F15 and Class. So, we cannot remove this feature.



1. Also, there are no outliers in feature F15. So, we are replacing missing values with ***‘Mean’***. (FYI: If there are outliers, we replace the missing values with Median).



1. As the and features are not scaled, we are ***‘Normalizing’*** them.

# Model Building & Hyper-parameter Tuning:

As the problem is classification, we are using ***‘Decision tree Classifier, Random Forest Classifier, K-Nearest Neighbors Classifier, Support Vector Machine classifier and XG-boost Classifier’.*** In any Machine Learning model there are various number of hyper parameters which need to be tuned according to the given data, so that the model can fit. So, we do hyper-parameter tuning using Cross Validation. We are using ***‘5-fold Grid Search Cross*** Validation’ which construct 5 versions of model with all the possible combinations of hyper-parameters. Taking consideration of best parameters, we have built learning models and calculated various evaluation metrics.

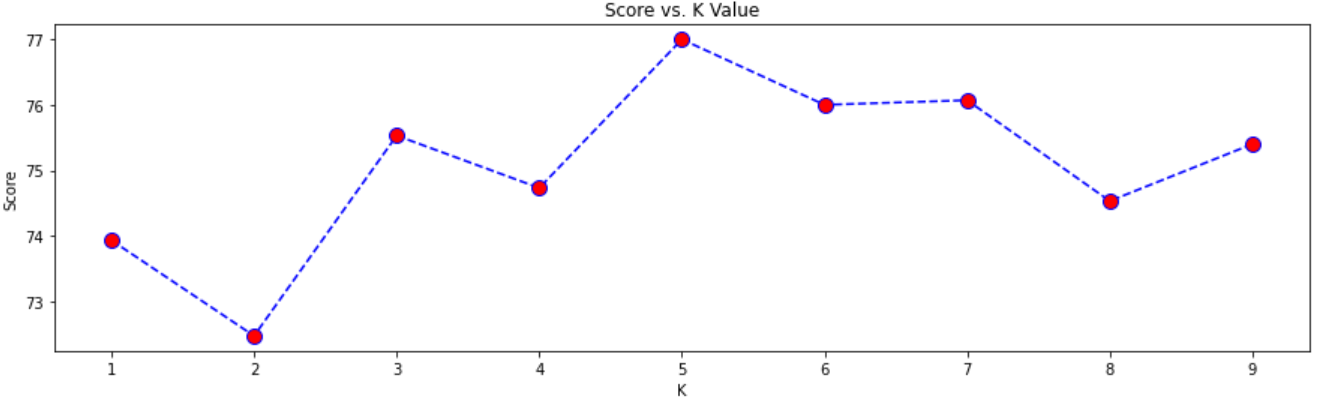
# Decision tree classifier:

The degree of understanding of the Decision Tree algorithm compared with other classification algorithms is so easy. The decision tree algorithm tries to solve the problem by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label. The hyper-parameters used for tuning are:

* ***‘Max\_depth’*** - Length of the longest path from the tree root to a leaf,
* ***‘Min\_samples\_split’*** - Minimum number of samples required to split an internal node and
* ***‘Min\_samples\_leaf’*** - Minimum number of samples required to split a leaf node.

# KNN classifier:

This is also one of the simplest algorithms in ML. It classifies data based on distance and number of nearest neighbors. So, it is the only parameter used for tuning. As you can see below when K is 5, we have the highest accuracy. So, we are taking ***‘n\_neighbors’*** as 5.



# Random Forest classifier:

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. Random Forest has nearly the same hyperparameters as a decision tree.

* ***‘Max\_depth’*** - Length of the longest path from the tree root to a leaf,
* ***‘Max\_features’*** - Number of features to consider when looking for the best split,
* ***‘Bootstrap’*** - Bootstrap sample is a random sample of observations, drawn with replacement,
* ***‘Criterion’*** - Gini, Entropy and
* ***‘Min\_samples\_split’*** - Minimum number of samples required to split an internal node are the parameters used for tuning.

# Support Vector Machine classifier (SVM):

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The hyper parameters used for tuning

* ***'C'*** - Soft margin cost function,
* ***'Gamma'*** - Free parameter of the Gaussian radial basis function,
* ***‘Kernel'*** - Rbf.

I am additionally using ***‘XG boost’*** just to check if it gives better accuracy.

# Model Evaluation:

We can make some quick comparisons between the different approaches used to improve performance showing the returns on each.

The following table shows the results from all the improvements we made:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1\_score | AUROC |
| Decision Tree | 80.8% | 79.2% | 79.2% | 0.792 | 0.789 |
| KNN | 77% | 76% | 75.9% | 0.757 | 0.753 |
| Random Forest | 85.87% | 85.9% | 85.9% | 0.859 | 0.857 |
| SVM | 90.4% | 90.1% | 90.1% | 0.901 | 0.90 |
| XG Boost | 89.07% | 89.5% | 89.5% | 0.895 | 0.894 |

Comparatively SVM is high in accuracy, F1\_score, Recall, Precision and AUROC. Henceforth, this learning model can be used as our final model.

# Prediction using best model:

Further step is to predict the output label "Class" in test data by using SVM learning model which gives whether the insured filed a claim or not.

**Part – 3**

# Data:

The Train dataset consists of historical data made of 1500 examples with 16 features starting from "F1" to "F16" and one output label "Target" representing the value of the claim. On the other hand, test dataset comprises of feature from "F1" to "F16" with missing output label class to be replaced with the output predictions.

# Pre-processing & Feature Engineering:

1. It is observed that the data contains categorical Features ‘F4’ and ‘F15’. So, I am converting them to numerical values using ***‘one-hot encoding’*** because many ML algorithms cannot work with categorical data directly.
2. As the and features are not scaled, we are ***‘Normalizing’*** them.

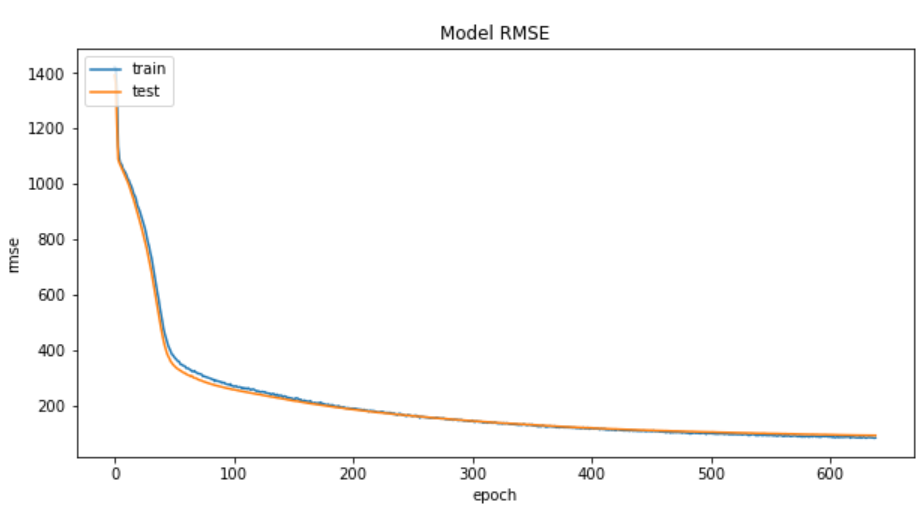
# Model Building & Hyper-parameter Tuning:

As the problem is Regression, apart from ***‘Linear Regression, KNN, Random Forest and SVM’***, I am also using ***‘Deep Neural Network’*** to get better model (for low Mean Squared Error). For hyperparameter tuning we are using ***‘Randomized Search Cross Validation’*** instead of Grid Search Cross Validation here because it selects random combinations to train the model and we can set search iterations based on our time and resources.

# Deep Neural Network:

I used 3 dense layers of 256 neurons each.

* ***‘Normal weight initializer’*** is used to prevent layer activation outputs from exploding or vanishing during a forward pass through a deep neural network.
* ***‘RELU’*** activation function is used as training a deep network with RELU tends to converge much more quickly and reliably than training with other activation functions like sigmoid.
* ***‘Adam Optimizer’*** is used as it changes the learning rate and momentum depending on the loss function. So, we do not need to extensively give them, and it is too fast and converges rapidly.
* Finally, ***‘Early stopping’*** is used as too many epochs can lead to overfitting of the training dataset, whereas too few may result in an underfit model. In the below graph you can clearly see the drop in train and test RMSE.



# Model Evaluation:

We can make some quick comparisons between the different approaches used to improve performance showing the returns on each. The following table shows the results from all the improvements we made.

|  |  |  |
| --- | --- | --- |
| Model | RMSE/MAE | MSE |
| Linear Regression | 389.5 | 249271.9 |
| KNN | 363.3 | 284573.2 |
| Random Forest | 267.8 | 139776.8 |
| SVM | 104.3 | 54600.9 |
| Deep Neural Network | 56.8 | 3235.7 |

Comparatively, ***‘Deep Neural Network’*** has low MSE and RMSE. Henceforth, this learning model can be used as our final model.

# Prediction using best model:

Further step is to predict the output label ***‘Target’*** in test data by using Deep Neural Network learning model which gives the value of the claim.

# References:

1. [www.kaggle.com](http://www.kaggle.com)
2. <https://scikit-learn.org/stable/>
3. <https://github.com/>
4. <https://machinelearningmastery.com/>
5. <https://towardsdatascience.com/>

1. <https://stackoverflow.com/>

1. <https://medium.com/>

1. <https://en.wikipedia.org/>