Miner2 Username: Sravv
Mason UserID: ssangara
Best public score: 0.63
Rank: 472

HW 2 : CREDIT RISK PREDICTION

Goal:

The objective of this assignment is to predict the credit risk of a person as high risk (0) or low risk (1) by developing a binary classification model which is trained using the given training data.

Approach:

I have implemented this assignment in two parts and the following files contain code as described below:

1) **HW2_ModelSelection.py**
   This file contains various classification models which have been used to calculate the accuracy based on F1 Score metric by splitting the training data using train_test_split() with test_size = 0.25. I have first preprocessed the data and then tried the following classification models and got the respective F1 scores.

Classification Model	**F1-Score**
Decision Tree	0.6258974993810350
K-Nearest Neighbors	0.5766202301635373
Linear SVM	0.4156226971260132
Kernel SVM	0.6246734397677793
Logistic Regression	0.5382904353352399
Gaussian Naïve Bayes	0.5979212253829321
Random Forest	0.6555883141248996

2) **HW2_Implemetation.py**
   In this file, I have implemented Random Forest Classification as it achieved the highest F1 Score compared to all the other classifiers that I have tried during model selection. I preprocessed the entire training data and trained the Random Forest classifier model using it. Then, predicted the credit risk of the given test data and got a public score of 0.63 on Miner2.

Data-preprocessing:

The first step was to import the libraries such as sklearn, pandas and matplotlib. Then I loaded the training data using read_csv() class from pandas library which was used to split the dataframe into two parts – one with all the features (X) and the other with the class label (y). Since the data had certain categorical features and some sensitive data among the independent variables, I had used One-Hot Encoding which replaces categorical column with new encoded columns using OneHotEncoder() from sklearn.preprocessing

module. The next step was to split the dataframe into training and test sets which was done using train_test_split() from sklearn.model_selection module with a test_size parameter as 0.25.

Then, I did feature scaling using StandardScaler() class from sklearn.preprocessing module which implements standardization on the data in order to put all the features on the same scale. This improves the model further as it doesn't let some features to dominate over the rest. Standardization works better than Normalization even if the data is not normally distributed and it scales the data in the range of [-3, +3].

The Standardization is calculated as below:

$$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$$

Model Selection:

In this step, I have tried various classification algorithms as mentioned at the start of this report on the preprocessed training set split and selected the best possible model with highest F1 score metric after validating with test set split. I also computed the confusion matrix using confusion_matrix() class from sklearn.metrics module.

The confusion matrix can be represented as following:

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

This is further used to calculate precision and recall, based on which F1- Score can be generated.

$$\text{Precision} = \frac{\text{True Positive}}{(\text{True Positive} + \text{False Positive})} \quad \text{Recall} = \frac{\text{True Positive}}{(\text{True Positive} + \text{False Negative})}$$

$$\text{F1-score} = \left(\frac{\text{Recall}^{-1} + \text{Precision}^{-1}}{2}\right)^{-1} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})}$$

Models Used:

1) **Decision Tree –** I imported DecisionTreeClassifier() class from sklear.tree module and gave the criterion parameter as “gini” so that the node split is done based on gini index.
2) **K Nearest Neighbors –** I used KNeighborsClassifier() class from sklearn.neighbors module and selected the parameters: n_neighbors = 100 which selects 100 nearest neighbors, and metric = 'minkowski', p = 2 which together calculates the Euclidean distance between the data points.
3) **SVM –** In order to use both linear and kernel SVM models, I imported SVC() class from sklearn.svm module and selected the parameters as kernel = 'linear’ for linear SVM model and kernel = 'rbf' for kernel SVM model.
4) **Logistic Regression –** I imported LogisticRegression() class from sklearn.linear_model module.
5) **Gaussian Naïve Bayes –** I had used GaussianNB() class from sklearn.naive_bayes module.
6) **Random Forest –** Since it is an ensemble model, I imported RandomForestClassifier() class from sklearn.ensemble module and selected the parameters as n_estimators = 300 which selects 300 decision trees, and criterion = 'gini' which is used to split the nodes based on gini index.

Then, all of these models were trained using the fit() method which takes matrix of features and class label as parameters. After the model is trained, predict() method is used on test data to predict the class label, which gives the credit risk.

Implementation:

As the highest F1-Score was achieved by Random Forest classifier when the parameter n_estimators was given as 300, I trained the model again using the entire training set after performing one hot encoding and feature scaling. Here, I used the gini index for splitting the nodes.

$$Gini = 1 - \sum_j p_j^2$$

References:

https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html
https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/
https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/
https://scikit-learn.org/stable/modules/classes.html