Credit Risk Prediction Assignment

Name: Yaswanth Reddy Manukonda
Email: ymanukon@gmu.edu
Miner2 Id: ymanukon
Mason Id: ymanukon
F1 Score: 68
Rank: 58

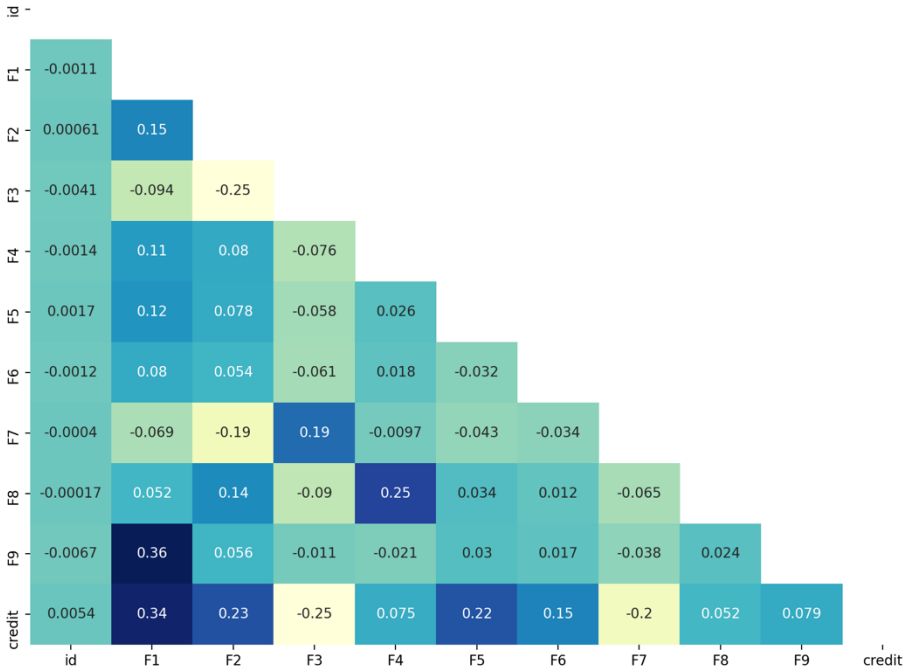
Problem Statement: develop a predictive model that can determine given a particular individual whether their credit risk is high denoted by 0 or low denoted by1.

Solution:

1. Preprocess the given reviews
2. Feature selection using correlation between each of them
3. Applying the classification model
4. Validate the F1 Score of the model using cross validation technique with the help of train data

Explanation:

- For preprocessing, I have followed the below steps
    1. Checked for missing values in the dataset and removed them, but it is observed that the given dataset doesn't have any null values in them
    2. Converted the given categorical features F10 and F11 to indicator values using **get_dummies** method in pandas and replaced the raw training data with them
- For feature selection I have calculated the correlation between all the features and found that the feature **id** has the low correlation with every other feature. So decided to remove the column **id** from the given datasets. Below is the plot that I have made to decide on the features. The implementation of this correlation can be found in **featureCorrelation.py**



- Now that I have preprocessed the data and removed the redundant features, to decide on the classifier algorithm with highest F1-score I have used 10 fold cross validation on the training dataset and found that ***GradiantBoostingclassifier*** is giving the highest average F1-Score of 0.68 on the train dataset splits. Below are all the classifiers and their parameters I have used
    1. ***KNeighborsClassifier:*** I have used all the default parameters and obtained an F1-score of 0.66. Implementation for this classifier is available in **KNN.py** file
        - n_neighbors =5 → number of neighbors to consider
        - weights = 'uniform' → all points in each neighborhood are weighted equally
        - metric = 'minkowski' → this is a distance metric
    2. ***GaussianNB:*** I have used Gaussian Naive Bayes with default parameters and got an average F1-score of 0.46 on 10 fold cross validation
    3. ***svm.LinearSVC:*** Linear Support Vector Classification has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples. This class supports both dense and sparse input. The Implementation of this classifier is available in ***SVM.py***
        - random_state: Controls the pseudo random number generation for shuffling the data
    4. ***ExtraTreesClassifier:*** I have used this as it is extremely randomized classifier compared to ***RandomForestClassifier*** which does the optimal splits and found an increase in F1-Score by 0.2
        - ***n_estimators:*** number of trees in the forest**.** I choose this parameter as 1000 and got better results
        - random_state: Controls the pseudo random number generation for shuffling the data
    5. ***GradientBoostingClassifier:*** builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. It has given me F1-Score of 0.68 with just default parameters
    6. ***DecisionTreeClassifier:*** to handle the imbalanced data, I have used the **class_weight** parameter. The implementation of ***DecisionTreeClassifier*** and its cross validation is available in the file ***DecisionTree.py***
        - ***class_weight:*** Weights associated with classes in the form *{class_label: weight}* this can be calculated with the below formula
            - $\boldsymbol{class\ weight\ of\ class1 = \frac{n_{samples}}{(n_{classes} * n_{samples\ of\ class1})}}$
        - ***max_depth:*** maximum depth of tree growth
        - ***min_samples_split:*** it is an integer that tells the classifier to split if there are more than specified values in any node
        - both the ***max_depth*** and ***min_samples_split*** are used to pre-prune the decision tree
        - however, I noticed GradientBoostingClassifier did better in terms of performance and accuracy compared to all other classifiers

- I have tried oversampling by adding the replica data, under sampling by deleting some data for all the classifiers. But couldn't see any significant increase in the F1Score during cross validation of those classifiers
- However, I see a significant increase in the F1-Score in DecisionTrees when I tried to balance the tree using class-weight parmeter

References:

- *API reference*. scikit. (n.d.). Retrieved September 24, 2021, from https://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble.
- *How to dealing with imbalanced classes in machine learning. Analytics Vidhya. (2021, January 6). Retrieved September 24, 2021, from https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/.*
- *sklearn.neighbors.KNeighborsClassifier. scikit. (n.d.). Retrieved September 28, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.*