Credit Risk Prediction

REQUIRED INFORMATION

Mason User Id: jbidhan Miner2 Username: flash

F1 Score and Rank: 0.68 , 20 (when submitted)

DESCRIPTION OF APPROACH:

I approached the problem reading the question and making notes for the steps required. The approach I used can be summarized as below:

1. **Reading and storing the Training file** using pandas.read_csv : I created 2 methods (function: readTrainfile() and readtestfile()) which needed a parameter for filepath to read the input. Using pandas.read_csv() function read the file and stored as a pandas' DataFrame having column names: ['UID', 'YEARS_TO_LAST_DEGREE', 'WORK_HOURS_PER_WEEK', 'CAT_VALUE', 'CAT_VALUE_OCCUPATION', 'GAINS', 'LOSS', 'MARITAL_STATUS', 'TYPE_EMPLOYMENT', 'TYPE_EDUCATION', 'TYPE_RACE', 'TYPE_GENDER', 'CREDIT_RISK'] and readtestfile() without the 'CREDIT_RISK' column and after input checked if any column has all null values then drop the column in the training set.
2. **Preprocessing of the data**: I created clean_data(data) method to process the data, where I dropped the non-value adding columns, performed one hot encoding via a modular function ONE_HOT_ENCODING() which removes the original column after performing pandas.get_dummies() to convert categorical variable to binary categorical data. I also removed the gender and added isFemale: 1 or 0 i.e., if the person is female then 1 otherwise 0. I also tried checking correlation between all the dataset and did not find any such strong relation.
3. **Cross Validation**: Dividing the training data using StratifiedKFold (sklearn.model_selection.StratifiedKFold).
4. **Prediction**: I created a common function evaluate_training_algorithm() which accepts data, n_folds and generates the folds of data for cross validation using a function called cross_validation() splits the data and returns the data using StratifiedKFold and for each fold of data calls Classification_Algo_Training() which accepts algorithm parameter, algorithm name and data set. The training data is passed through SMOTENC i.e. Synthetic Minority Over-sampling Technique for Nominal and Continuous from the imblearn.over_sampling class. It is used on the dataset containing numerical and categorical feature using the index stored after preprocessing of the data. The prediction step returns the Predictions and the F1 Score. The classifiers used are:
    - from sklearn.ensemble => RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
    - from sklearn.linear_model => LogisticRegression
    - from sklearn.naive_bayes => GaussianNB
    - from sklearn.svm => LinearSVC
    - from sklearn.neighbors => KNeighborsClassifier (various Values of K)

    I tried applying various parameters like max_depth ,max_features, n_neighbors, random_state, bootstrap, etc in the above algorithms separately. (not all together but depending on each class's parameters)
5. **Finding accuracy**: The accuracy is calculated for each prediction via the F1 Score i.e the weighted average of Precision and Recall is the F1 Score using the sklearn.metrics class 's f1_score.
6. **Real Test and Prediction:** Reading test data file and applying Steps 1,2 and 4. Then using the saveOutput() store the output in the txt file and upload on the miner portal to get the results.

*The library modules which were used were wrapped for modularity and the sake of reusability.

Methodology of Choosing the Approach and Associated Parameters and Graphs

I started writing the code on Jupyter notebook and created two files, one for analysis and other for actual prediction. I tried running the data directly without cleaning, the accuracy was not great and was around 0.37 and then with categorical data's processing it reached around 0.64 using various classifiers. Then tuned the parameter of parameters like max_Depth and max_features(log2,None, Auto, Sqrt). There were several factors involved in choosing the approach which are answered in the following questions:

1. How did you deal with different features?
    a. **Types of Features:** Categorical and Continuous columns were present in the data.

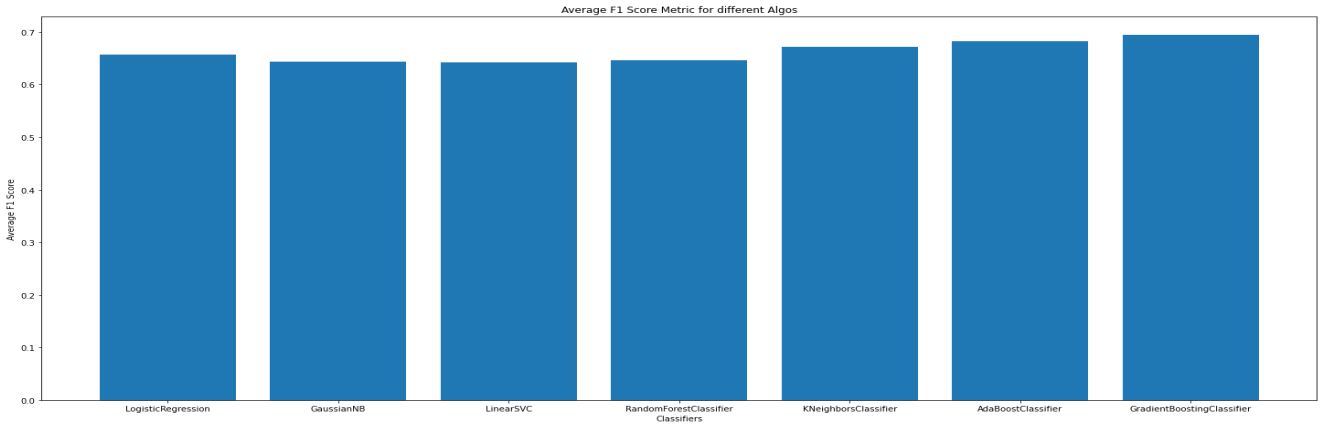
b. **Preprocessing of Data:** Remove the empty columns and rows from the training data. Also dropped UID as it was carrying no helpful information. I was not able to find strong correlation between the data.

	UID	YEARS_TO_LAST_DEGREE	WORK_HOURS_PER_WEEK	CAT_VALUE	CAT_VALUE_OCCUPATION	GAINS	LOSS	MARITAL_STATUS	TYPE_EMPLOYMENT	TYPE_EDUCATION	CREDIT_RISK
UID	1.000000	-0.001019	0.000605	-0.004070	-0.001500	0.001680	-0.001200	-0.000352	-0.000051	-0.006715	0.005374
YEARS_TO_LAST_DEGREE	-0.001019	1.000000	0.148127	-0.094145	0.109748	0.122627	0.079932	-0.069338	0.052015	0.359172	0.335182
WORK_HOURS_PER_WEEK	0.000605	0.148127	1.000000	-0.248975	0.080384	0.078409	0.054256	-0.190521	0.138974	0.055510	0.229690
CAT_VALUE	-0.004070	-0.094145	-0.248975	1.000000	-0.075620	-0.057918	-0.061064	0.185461	-0.090449	-0.010879	-0.250924
CAT_VALUE_OCCUPATION	-0.001500	0.109748	0.080384	-0.075620	1.000000	0.025511	0.017979	-0.009617	0.255004	-0.021274	0.075448
GAINS	0.001680	0.122627	0.078409	-0.057918	0.025511	1.000000	-0.031614	-0.043398	0.033828	0.030047	0.223333
LOSS	-0.001200	0.079932	0.054256	-0.061064	0.017979	-0.031614	1.000000	-0.034182	0.012231	0.016744	0.150523
MARITAL_STATUS	-0.000352	-0.069338	-0.190521	0.185461	-0.009617	-0.043398	-0.034182	1.000000	-0.064798	-0.038398	-0.199295
TYPE_EMPLOYMENT	-0.000051	0.052015	0.138974	-0.090449	0.255004	0.033828	0.012231	-0.064798	1.000000	0.023537	0.051645
TYPE_EDUCATION	-0.006715	0.359172	0.055510	-0.010879	-0.021274	0.030047	0.016744	-0.038398	0.023537	1.000000	0.079311
CREDIT_RISK	0.005374	0.335182	0.229690	-0.250924	0.075448	0.223333	0.150523	-0.199295	0.051645	0.079311	1.000000

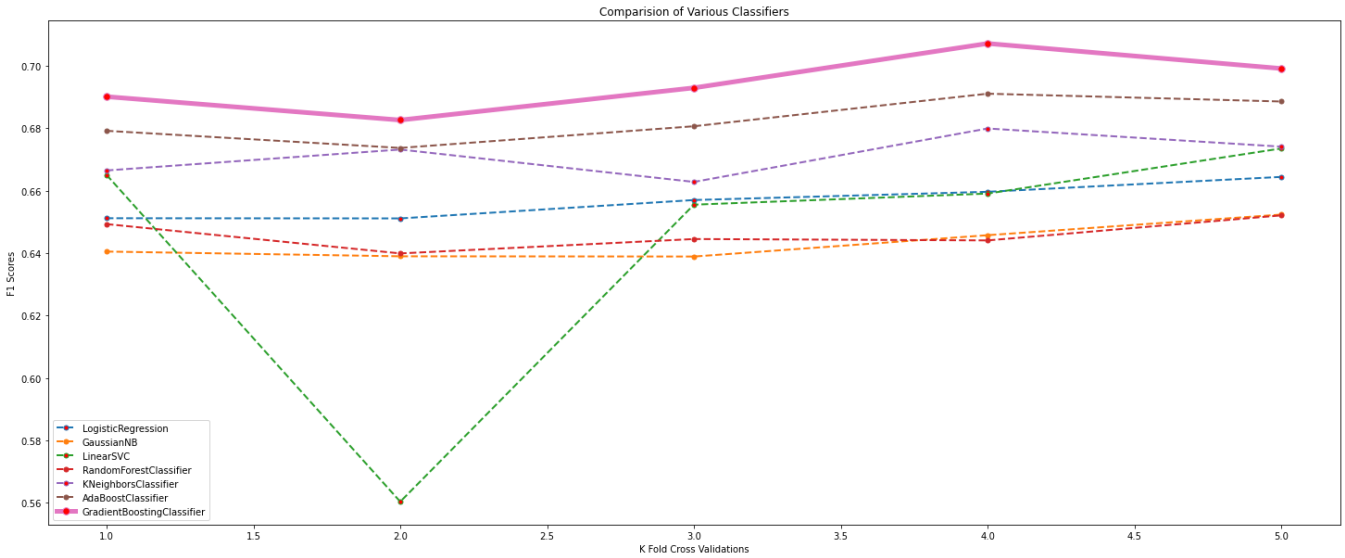
c. **One Hot Encoding:** Using the pandas.get_dummies() generated Boolean categorical data and remove the original categorical columns. I also removed the Gender column and replaced it with IsFemale, i.e if person is Female then 1 otherwise 0.

d. **Normalization of the Continuous data:** I Used Robust Scaler algorithm as it scales the data, and it is robust to the outliers. It uses the interquartile range. The median and scales of the data are removed by this scaling algorithm according to the quantile range. I tried other normalization techniques but those did not provide improvement in the F1 Score.

2. **Did you exclude any specific features?**
   Yes, I dropped off the UID as it was unique for all the rows.
3. **Was there a certain way you dealt with imbalance in the class distributions?**
   a. **Cross Validation:** I decided to use StratifiedKFold to split data into 5 folds for better validation. I then created sets out of data using a modular function. I used StratifiedKFold to ensure that each fold of dataset has the same proportion of observations with a given label and the datasets are balanced.
   b. **Synthetic Minority Over-sampling Technique for Nominal and Continuous (SMOTENC):** from the imbalanced-learn library, which creates synthetic data for categorical as well as quantitative features in the data set. SMOTENC slightly changes the way a new sample is generated by performing something specific for the categorical features. SMOTENC is a technique based on nearest neighbors judged by Euclidean Distance between data points in feature space. On adding it, there was increase in accuracy from 0.66 to 0.68.
4. **How did you perform model selection and which classifier stood out? Any theoretical reasoning why?**
   I performed model selection based on average F1 scores on the various model checked on various experiments performed over the different classifier algorithms. The below bar graph shows the average accuracies of each of the classifier I tried. The data for them is as follows:
   i. LogisticRegression : 0.6566836305022689
   ii. GaussianNB : 0.6432987595066393
   iii. LinearSVC : 0.6427111921121552
   iv. RandomForestClassifier : 0.6459786591009992
   v. KNeighborsClassifier : 0.6713221272228602
   vi. AdaBoostClassifier: 0.6826596941113323
   vii. **GradientBoostingClassifier : 0.6944315486926854 (Best and it remained constant for all the iterations while performing cross validation)**



F1 Score: As the class distribution is unequal F1 is typically more useful than accuracy. So, I used F1 score to compare the classifiers I.e. The weighted average of Precision and Recall is the F1 Score. As a result, this score considers both false positives and false negatives. Although it is not as intuitive as accuracy. Below is the plot for F1 Score for various K-Fold. Here we observed that F1 Score remained high for Gradient Boosting Classifier.



Gradient boosting is a greedy algorithm and is one of the Arching algorithms. Boosting refers to this general problem of producing a very accurate prediction rule by combining rough and moderately inaccurate rules-of-thumb. Arcing is an acronym for Adaptive Reweighting and Combining. Each step in an arcing algorithm consists of a weighted minimization followed by a recomputation of [the classifiers] and [weighted input]. The statistical framework cast boosting as a numerical optimization problem where the objective is to minimize the loss of the model by adding weak learners using a gradient descent like procedure. This class of algorithms were described as a stage-wise additive model. This is because one new weak learner is added at a time and existing weak learners in the model are frozen and left unchanged. We generally use Gradient Boosting Algorithm when we want to decrease the Bias error. Gradient boosting algorithm is one of the powerful algorithms that can be used for predicting categorical target variable (as a Classifier), the cost function is Log loss. I tried other classifier experiments but none of them reached near AdaBoost or Gradient Boosting Algorithms. Logistic Regression, K-Neighbours (Tired for various values of K) and RandomForestClassifier were closely following but did not touch .70 F1 score.

CONCLUSION

On seeing the results of the various experiments on various classifiers, it was thus decided to go with the Gradient boosting as it turned out to give the best F1 score with the constantly high F1 scores while cross validation. Without SMOTENC the accuracy was 0.66 but on adding SMOTENC, the accuracy increased till .70 on the training set while achieved .68 on the miner portal.

REFERENCES:

Documentation from https://scikit-learn.org/ , https://www.nltk.org/, https://www.scipy.org , https://matplotlib.org/ , and https://imbalanced-learn.org/stable/over_sampling.html

Blogs: https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/ , https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/,