CREDIT RISK PREDICTION

Name: Hasmitha Bhutham
GNumber: G01205552
Username on Miner: acchickens
Rank :351, Score on Miner: 0.65

PROGRAM IMPLEMENTATION:

Main steps include:

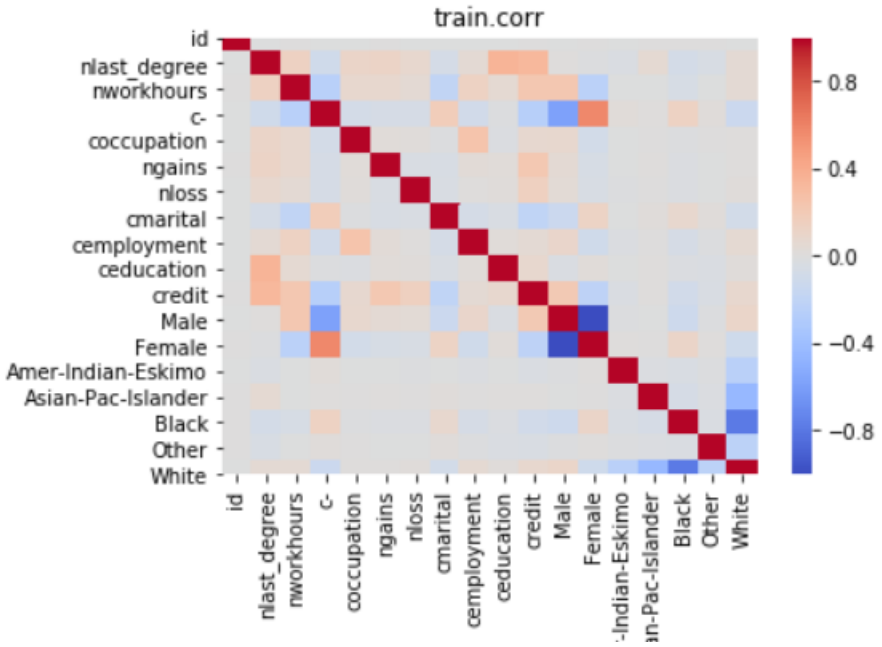
1. Reading and naming the data.
2. Encoding using one-hot encoder.
3. Finding the correlation.
4. Splitting the target variable & resampling the data.
5. DecisionTreeClassifier and BaggingClassifier.
6. Random Forest Classifier.
7. Prediction and finding f1_score.

1. **Reading and Naming the data:**
   - The data files have been read using pandas read_csv().
   - The columns have been renamed as:

```
train.columns = ['id', 'nlast_degree', 'nworkhours', 'c-',
'coccupation', 'ngains', 'nloss', 'cmarital', 'cemployment',
'ceducation', 'crace', 'cgender', 'credit']

test.columns = ['id', 'nlast_degree', 'nworkhours', 'c-',
'coccupation', 'ngains', 'nloss', 'cmarital', 'cemployment',
'ceducation', 'crace',
       'cgender']
```

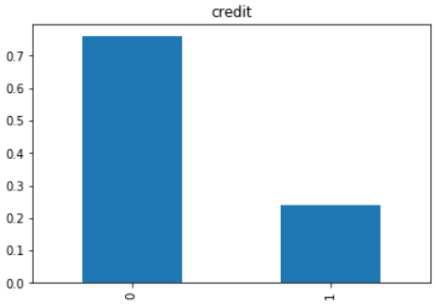
   to make visualization easier.
2. **Encoding using one-hot-encoder:**
   - One-hot encoder has been used on ‘cgender’ and ‘crace’.
   - get_dummies() method has been used to encode the categorical values of the train and test data.
3. **Finding the correlation:**
   - The correlation between the columns has been found using a heatmap as below which shows c- has least feature importance so the column has been dropped.

```
n [92]:  #heatmap of features
         sns.heatmap(train.corr(),cmap = 'coolwarm')
         plt.title('train.corr')

Out[92]: Text(0.5, 1, 'train.corr')
```



4. **Splitting the target variable & resampling the data:**
   - The target variable has been split into two classes for 0s and 1s respectively.
   - Since there is unbalanced data as shown in the graph below, resampling has been done.

```
In [68]:  #using value_counts() to analyze the 0s and 1s
          train["credit"].value_counts(normalize=True).plot.bar(ti

Out[68]: <matplotlib.axes._subplots.AxesSubplot at 0x219286bb7c8>
```



   - The resample() function is used here and is concatenated with the split data.
5. **DecisionTreeClassifier and BaggingClassifier:**
   - The BaggingClassifier is used to improve the predictions by taking in the predictions from multiple trees.

- The params are: `BaggingClassifier(base_estimator=DecisionTreeClassifier(class_weight=None, criterion='gini',max_depth=None,max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None,min_samples_leaf=1,min_samples_split=2, min_weight_fraction_leaf=0.0,presort=False, random_state=None,splitter='best'),bootstrap=True, bootstrap_features=False, max_features=1.0,max_samples=1.0, n_estimators=100, n_jobs=None, oob_score=False, random_state=7, verbose=0,warm_start=False)`
- The num of trees used here 100.

6. **Random Forest Classifier:**
    - Random Forest Classifier was also used to determine the better f1_score.
    - Params are: `RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',max_depth=None, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None,min_samples_leaf=1, min_samples_split=2,min_weight_fracti on_leaf=0.0, n_estimators=10, n_jobs=None, oob_score=False, random_state=None,verbose=0, warm_start=Fals)`
    - F1_score for Rfc is 0.64

7. **Prediction and F1_score:**
    - The prediction is done using ‘test2’ data and f1_score is determined for y_test2 and bp.
    - The f1_score is calculated as 0.670142 for BaggingClassifier and is uploaded on miner.
    - The result file is then printed in the same folder with the predictions.