Project Report, CMSC 6950

S.M. Amin Taheri G. - 202193514

June 16, 2022

Abstract

The purpose of this project to correctly identify banking customers that are going to continue with their current bank or they they want to close their account. This is a binary classification problem. The dataset is at here

1 Analysis Methods

1.1 K means clustering with Scipy

Over successive iterations, the k-means algorithm adjusts the clustering of observations and updates the cluster centroids until the centroids remain stable. By comparing the absolute value of the change in the average Euclidean distance between the observations and their corresponding centroids with a threshold, this algorithm implementation determines the stability of the centroids. As a result, a code book is generated, mapping centroids to codes and vice versa.

```
# cluster the data based on the feature values
from sklearn.metrics import accuracy_score
from scipy.cluster.vq import kmeans, vq
df_cluster = robust_scaler_df.copy() # copy the dataframe
results, _ = kmeans(df_cluster.values, 2)
codes, _ = vq(df_cluster.values, results)

# add the cluster column to the dataframe
codes = pd.DataFrame(codes, columns=['Cluster'])
df_cluster = df_cluster.join(codes)
df_cluster = df_cluster.join(df['Exited'])

# Drop the rows that contain nan values
df_cluster.dropna(inplace=True)

y_hat = df_cluster['Cluster']
y = df_cluster['Exited']

# accuracy of the model
print("Accuracy of the clustering: ", accuracy_score(y_hat, y))
```

1.2 Gradient Boosting Classifier

Gradient Boosting is an inductively generated tree ensemble method that builds a forward stepwise additive model. At each step, n_class regression trees are trained against the negative gradient of the binomial or multinomial deviance loss function. The grid options and the resulting best parameters are as follows:

```
gb_cv = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1)

# grid search for the best parameters
param_grid = {'max_depth': [3, 5, 7], 'min_samples_leaf': [1, 5, 10], 'max_features': [1.0, 0.3, 0.1]}
grid = GridSearchCV(gb_cv, param_grid, cv=5)
grid.fit(X, y)

# print the best parameters
print("Best parameters: ", grid.best_params_)

Best_params = {'max_depth': 5, 'max_features': 0.3, 'min_samples_leaf': 5}
gb_cv = GradientBoostingClassifier(**Best_params)
results = cross_val_score(gb_cv, X, y, cv=5)
print("Accuracy of Gradient Boosting Classifier: ", results.mean())
```

2 Performance

2.1 Metric

As this problem is a classification problem, the accuracy metric was chosen for this project. Also, the data-set was a balanced data-set meaning that the two classes had approximately the same number of each class. So, choosing a metric that is robust to imbalanced data was no necessary here. The formula of accuracy is

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

2.2 Statistics

As the table below shows, we evaluated two machine learning models, clustering with K-means and Gradient Boosting. Gradient Boosting algorithm outperforms k means with a noticeable distance.

Model	Accuracy
K-means	0.5126
Gradient Boosting	**0.861**

3 Acknowledgement

- Gradient boosting classifier documentation in Sklearn
- Different score functions in Sklearn
- Scipy clustering algorithms, K-means

4 Program Specification

- Python = 3.8.12
- Pandas = 1.3.5
- Scikit-learn = 1.0.2
- Numpy = 1.21.2
- Matplotlib = 3.5.0
- Scipy = 1.8.0
- Seaborn = 0.11.2
- pytables = 3.7.0 (This is for exporting to hd5 file)