Individual Final Report

DATS 6103 Final

Renping Ge

1. Introduction

A company which is active in Big Data and Data Science wants to hire data scientists among people who have passed some courses provided by company. Many people among them signup for their training. Company wants to find out which candidates really want to get the job after training because it would help them to reduce the cost and time, as well as the quality of training. Our project using this dataset from Kaggle aiming to predict who will move to a new job.

The dataset contains 19158 entries and 14 columns including features variables and target variable. And tasks for our project mainly have two parts, one for EAD and data preprocess, and one for model building and evaluation and GUI display.

Li took responsibility for most parts of EDA and data preprocess, and I'm in charge of most model parts and GUI parts.

2. Description of your individual work

My work for the final project could be divided into two parts: Models and GUI.

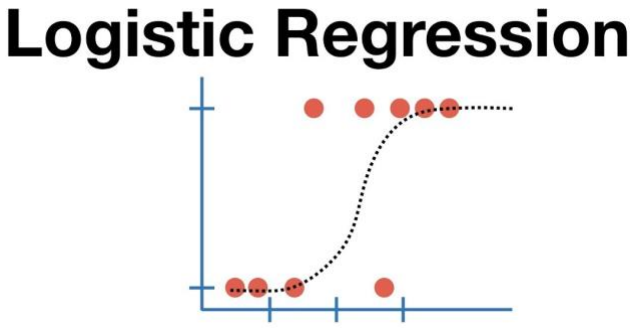
For the model part, I built up 2 models using logistic regression and gradient boosting.

2.1. Logistic Regression

Logistic regression is a statistical model that uses logistic function to model the conditional probability. It models the probabilities for binary classification question. The logistic function is defined as below:

$$logistic(\partial) = \frac{1}{1 + \exp(-\partial)}$$

And it also looks like below figure, transformed into the range 0 and 1 using logistic regression.



2.2. Gradient Boosting

Gradient boosting is an algorithm that could overfit a training dataset quickly. It gets benefit from regularization and generalizes item by allowing optimization of loss function. The equation of binary cross entropy loss is shown as below. If we have a random variable X, then the pdf of X would show as below:

$$s = \begin{cases} -\int p(x).logp(x)dx & if\ x\ is\ continuous \\ -\sum p(x).logp(x) & if\ x\ is\ discrete \end{cases}$$

In this case, the loss function for the dependent variable y would be:

$$L = -y * \log(p) - (1 - y) * \log(1 - p)$$

2.3. GUI

I created a GUI for model part, setting several buttons for different model. Each time we click on the corresponding button, the result would display in the window.

3. Describe the portion of the work that you did on the project in detail

In this part, I will describe my portion of work on the project more detailed. Below is the code for gradient boosting model built and evaluation, as well as the logistic regression. I have tried several other classifiers, like KNN and XGB boosting. However, their results are kind of poor compared to our main models selected.

```
# model part
# boosting model
from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import HistGradientBoostingClassifier
from sklearn.metrics import precision_recall_fscore_support
# fitting model
hgbc = HistGradientBoostingClassifier(random_state=42)
hgbc.fit(X_train, y_train)
# Get the prediction
y_valid_pred = hgbc.predict(X_test)
y_valid_pred_score = hgbc.predict_proba(X_test)
# Get the precision, recall and f-score
precision, recall, fscore, support = precision_recall_fscore_support(y_test,
y_valid_pred, average='micro')
# print our result
print(classification_report(y_test,y_valid_pred))
print("Accuracy:", accuracy_score(y_test, y_valid_pred) * 100)
```

```
print("ROC_AUC : ", roc_auc_score(y_test,y_valid_pred_score[:,1]) * 100)
# confusion matrix
conf_matrix = confusion_matrix(y_test, y_valid_pred)
df_cm = pd.DataFrame(conf_matrix, index=['0','1'], columns=['0','1'] )

plt.figure(figsize=(5,5))
hm = sns.heatmap(df_cm, cbar=False, annot=True, square=True, fmt='d',
annot_kws={'size': 20}, yticklabels=df_cm.columns, xticklabels=df_cm.columns)
hm.yaxis.set_ticklabels(hm.yaxis.get_ticklabels(), rotation=0, ha='right',
fontsize=20)
hm.xaxis.set_ticklabels(hm.xaxis.get_ticklabels(), rotation=0, ha='right',
fontsize=20)
plt.ylabel('True label',fontsize=20)
plt.xlabel('Predicted label',fontsize=20)
plt.tight_layout()
plt.show()
# Plot ROC Area Under Curve
fpr, tpr, _ = roc_curve(y_test, y_valid_pred_score[:,-1])
auc = roc_auc_score(y_test, y_valid_pred_score[:,-1])
plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange',
         lw=lw, label='ROC curve (area = %0.2f)' % auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curve')
plt.legend(loc="lower right")
plt.show()

#%%-------------------------------------------------------------------------------
----
## Modeling
### logistic regression
from sklearn.linear_model import LogisticRegression
# Fitting model
logt = LogisticRegression()
logt.fit(X_train,y_train)
# Get the prediction
y_valid_pred2 = logt.predict(X_test)
y_valid_pred2_score = logt.predict_proba(X_test)
```

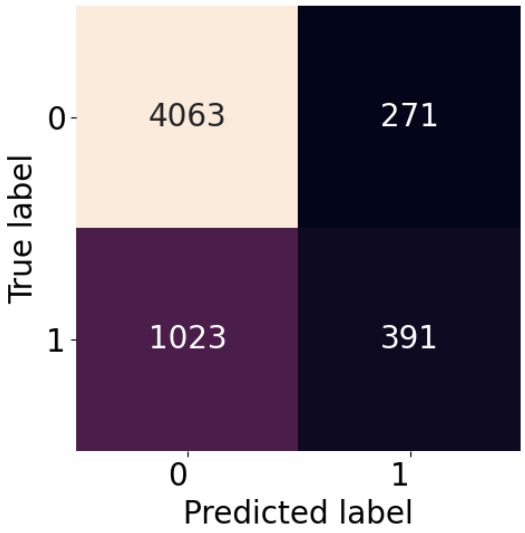
```
# Print result
print(classification_report(y_test,y_valid_pred2))
print("Accuracy : ", accuracy_score(y_test, y_valid_pred2) * 100)
print("ROC_AUC : ", roc_auc_score(y_test,y_valid_pred2_score[:,1]) * 100)
### confusion matrix
conf_matrix = confusion_matrix(y_test, y_valid_pred2)

df_cm = pd.DataFrame(conf_matrix, index=['0','1'], columns=['0','1'] )

plt.figure(figsize=(5,5))
hm = sns.heatmap(df_cm, cbar=False, annot=True, square=True, fmt='d',
annot_kws={'size': 20}, yticklabels=df_cm.columns, xticklabels=df_cm.columns)
hm.yaxis.set_ticklabels(hm.yaxis.get_ticklabels(), rotation=0, ha='right',
fontsize=20)
hm.xaxis.set_ticklabels(hm.xaxis.get_ticklabels(), rotation=0, ha='right',
fontsize=20)
plt.ylabel('True label',fontsize=20)
plt.xlabel('Predicted label',fontsize=20)
plt.tight_layout()
plt.show()

#%%------------------------------------------------------------------
# Plot ROC Area Under Curve
fpr, tpr, _ = roc_curve(y_test, y_valid_pred2_score[:,-1])
auc = roc_auc_score(y_test, y_valid_pred2_score[:,-1])

plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange',
         lw=lw, label='ROC curve (area = %0.2f)' % auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curve')
plt.legend(loc="lower right")
plt.show()
```

The GUI part code is very long, so I will not conclude them here. It will be shown both in the code part and my_work.py. However, I think we still need to improve our GUI code.

4. Results

4.1. Logistic Regression Result

From the confusion matrix, we could find that there are 4063 true negatives, and 391 true positives, the result is shown as below:



From the classification report graph below, we can see the accuracy of logistic regression model is 77.48%. It means, for the given test data set, the ratio of the number of samples correctly classified by the classifier to the total number of samples is 77.48%. It is calculated by (TP+TN)/(TP+FP+FN+TN). Here, TP represents the true positive, and the FP represents the false positive, and the FN represents the false negative, and TN means the true negative. We could get those result from confusion matrix above.

	precision	recall	f1-score	support
0	0.80	0.94	0.86	4334
1	0.59	0.28	0.38	1414
accuracy			0.77	5748
macro avg	0.69	0.61	0.62	5748
weighted avg	0.75	0.77	0.74	5748

Accuracy : 77.48782185107864
ROC_AUC : 78.41463080318185

The precision here means the ratio of true positive to the sum of true positive and false positive. And the recall means the ratio of true positive to the sum of true positive and

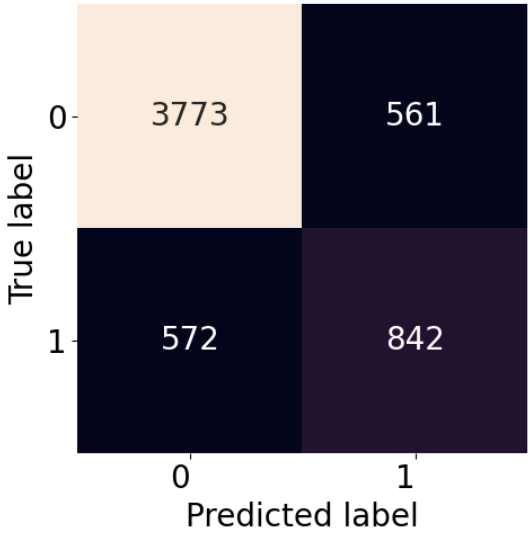
false negative. F-1 score is a combination index, it means the harmonic mean of precision and recall. The equation is like:

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

The AUC means the area under the ROC curve. It shows the tradeoff of specificity and sensitivity. And roc_auc sore helps us find the model performances. Normally, the roc_auc score higher than 0.5 would see as a good classifier. I would combine there different index in our summary to interpret the result.

4.2. Gradient Boosting Result

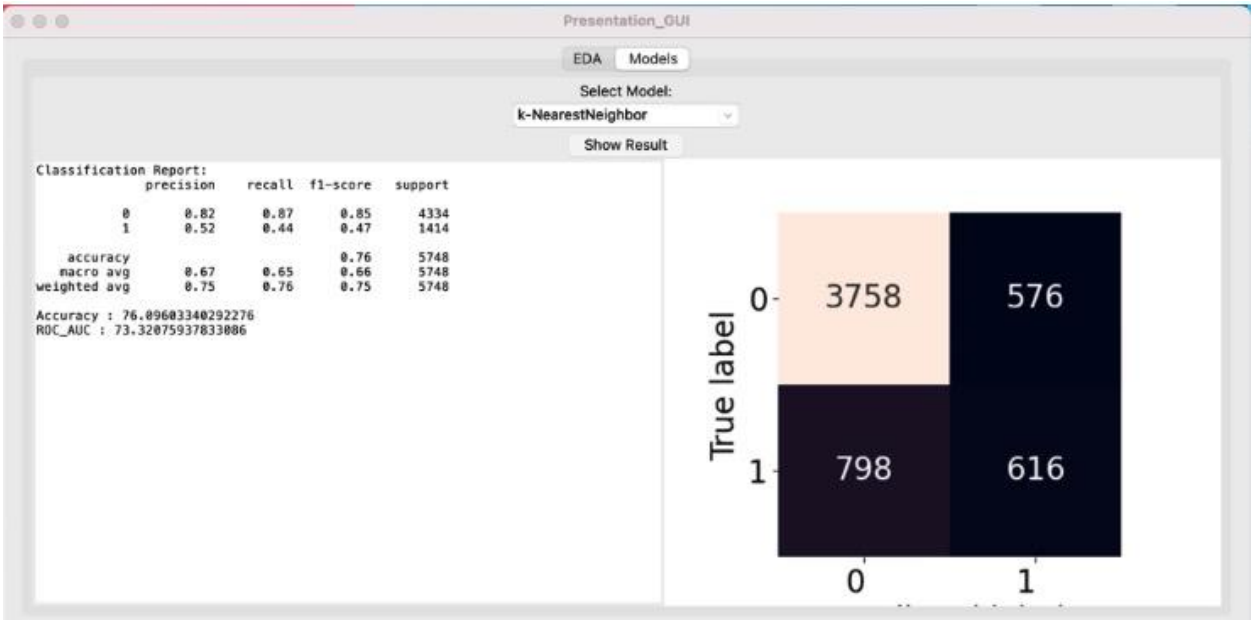
From the gradient boosting result, we could see that there are 3663 true negatives and 842 true positives. The result is shown as below.



The accuracy for gradient boosting model is 80.28%, which means that the fitting model could explain 80.28% of the test dataset.

4.3. GUI

Finally, I tried to implement GUI. We divided it into two parts, EDA part and Model part. I built up the model part. By clicking the different model options, we could directly get the result of our classification model.



5. Summary and conclusions

Based on our results and the index we have discussed before; we could define that gradient boosting is the best model. So, we choose it to do prediction. This model has 80.28% accuracy and ROC_AUC score is 0.80, which means it could explain the 80.28% of the test data. And it has the highest ROC_AUC score.

Besides the models we talked about, I also conduct several other classifiers, like XGB boosting, KNN, etc.… But none of them has the better result than the models we selected.

The improvement of our project may mainly implement in GUI part and data preprocess. Maybe trying another way to deal with our data would have a better result.

6. Calculate the percentage of the code that you found or copied from the internet

$$\frac{400 - 180}{400 + 110} = 43.1\%$$

7. References

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. Frontiers in neurorobotics, 7, 21.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. Advances in neural information processing systems, 30, 3146-3154.

Menard, S. (2002). Applied logistic regression analysis (Vol. 106). Sage

https://www.kaggle.com/arashnic/hr-analytics-job-change-of-data-scientists

https://machinelearningmastery.com/logistic-regression-for-machine-learning/

https://christophm.github.io/interpretable-ml-book/logistic.html