1. Introduction.

Through analysis of dataset on prediction of job change, I sharped the skills learned on the Data Mining course, including data preprocessing, data encoding, exploratory data analysis, modeling building and model evaluation.

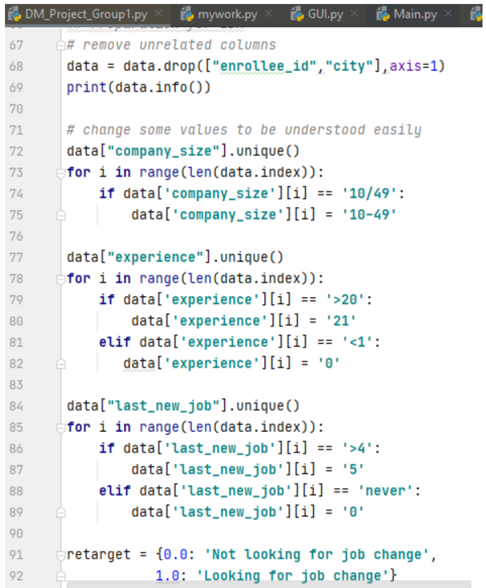
2. Description of your individual work.

My work on this project consists the following part:

1)dataset loading 2)data preprocessing 3)exploratory data analysis 4)data encoding 5)model building 6)model evaluation 7) final report-EDA and Appendix. The model I used on this dataset is KNN and Random Forest.

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

1) Data preprocessing



DM_Project_Group1.py mywork.py GUI.py Main.py

```
# remove unrelated columns
data = data.drop(["enrollee_id","city"],axis=1)
print(data.info())

# change some values to be understood easily
data["company_size"].unique()
for i in range(len(data.index)):
    if data['company_size'][i] == '10/49':
        data['company_size'][i] = '10-49'

data["experience"].unique()
for i in range(len(data.index)):
    if data['experience'][i] == '>20':
        data['experience'][i] = '21'
    elif data['experience'][i] == '<1':
        data['experience'][i] = '0'

data["last_new_job"].unique()
for i in range(len(data.index)):
    if data['last_new_job'][i] == '>4':
        data['last_new_job'][i] = '5'
    elif data['last_new_job'][i] == 'never':
        data['last_new_job'][i] = '0'

retarget = {0.0: 'Not looking for job change',
            1.0: 'Looking for job change'}
```

2) exploratory data analysis

```
## EDA
# show counts for target
target = data.groupby('target').agg({'target': 'count'}).rename(columns =
{'target': 'count'}).reset_index()
a = sns.barplot(data = target,x = target['target'], y = target['count'])
for p in a.patches:
    percentage = '{:.1f}%'.format(100 * p.get_height()/len(data.target))
    x = p.get_x() + p.get_width() / 2 -0.1
    y = p.get_y() + p.get_height()
    a.annotate(percentage, (x, y), size = 12)
plt.title('target', size = 16)
plt.show()

# Distribution of job change by gender
gender_df = data.groupby(['gender', 'target']).agg({'target':
'count'}).rename(columns = {'target': 'count'}).reset_index()
# genderdf_agg = genderdf.groupby(['gender'])['count'].sum().reset_index()
# genderdf2 = genderdf.merge(genderdf_agg, on='gender', how='left')
# genderdf2['percentage']=round(genderdf2.count_x/genderdf2.count_y * 100,1)
b = sns.barplot(data = gender_df, x = gender_df['gender'], y =
gender_df['count'],hue = gender_df['target'])

patch_height = [p.get_height() for p in b.patches]
patch = [p for p in b.patches]
for i in range(gender_df["gender"].unique().size):
    total = gender_df.groupby(['gender'])['count'].sum().values[i]
    for j in range(gender_df["target"].unique().size):
        percentage = '{:.1f}%'.format(100 * patch_height[(j *
gender_df["gender"].unique().size+i)]/total)
        x = patch[(j * gender_df["gender"].unique().size+i)].get_x() +
patch[(j * gender_df["gender"].unique().size+i)].get_width() / 2 -0.1
        y = patch[(j * gender_df["gender"].unique().size+i)].get_y() +
```

```
patch[(j * gender_df["gender"].unique().size+i)].get_height()
        b.annotate(percentage, (x, y), size = 12)
plt.title('gender', size = 16)
plt.show()
```

3) data encoding

```
X = data.drop(["target"],axis = 1)
y = data["target"]

# X = data.values[:, 0:11]
# y = data.values[:, 11]

# fill na
print("Sum of NULL values in each column. ")
print(data.isnull().sum())
X['experience'] = X['experience'].astype('float64').fillna(X['experience'].mean())
X['last_new_job'] = X['last_new_job'].astype('float64').fillna(X['last_new_job'].mean())
X['training_hours'] = X['training_hours'].astype('float64').fillna(X['training_hours'].mean())

# # standerization and centralization
# X.dropna(how='any')
sc = StandardScaler()
X["city_development_index"] = sc.fit_transform(X["city_development_index"].values.reshape(-1,1))
X["experience"] = sc.fit_transform(X["experience"].values.reshape(-1,1))
X["last_new_job"] = sc.fit_transform(X["last_new_job"].values.reshape(-1,1))
X["training_hours"] = sc.fit_transform(X["training_hours"].values.reshape(-1,1))

# encoding categorical features with OneHotEncoder()
columns_categorical = ["gender","relevent_experience","enrolled_university","education_level","major_discipline","company_size","company_
columns_numerical = ["city_development_index","experience","last_new_job","training_hours"]

X = pd.get_dummies(X, columns = columns_categorical)

# label target variable
le = LabelEncoder()
y = le.fit_transform(y)

# split the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=2000)
```

4) modeling with KNN and Random Forest

```
#%%-----------------------------------------------------------------------
## Modeling
# perform training with random forest with all columns
# specify random forest classifier and perform training
clf = RandomForestClassifier(n_estimators=90)
clf.fit(X_train, y_train)

#%%-----------------
# get feature importances
importances = clf.feature_importances_

# convert the importances into one-dimensional 1darray with corresponding df column names as axis labels
f_importances = pd.Series(importances, X.columns)

# sort the array in descending order of the importances
f_importances.sort_values(ascending=False, inplace=True)

# make the bar Plot from f_importances
f_importances.plot(x='Features', y='Importance', kind='bar', figsize=(16, 9), rot=90, fontsize=15)

# show the plot
plt.tight_layout()
plt.show()
#%%-----------------------------------------------------------------------
## Make predictions
y_pred = clf.predict(X_test)
y_pred_score = clf.predict_proba(X_test)
```

```
# #%%-----------------------------------------------------------------------
# ## KNN
#
# clf_KNN = KNeighborsClassifier(n_neighbors=5)
# clf_KNN.fit(X_train, y_train)
#
# y_pred_KNN = clf_KNN.predict(X_test)
#
# #%%-----------------------------------------------------------------------
# # calculate metrics
#
# print("\n")
# print("Classification Report: ")
# print(classification_report(y_test,y_pred_KNN))
# print("\n")
# print("Accuracy : ", accuracy_score(y_test, y_pred_KNN) * 100)
# print("\n")
```

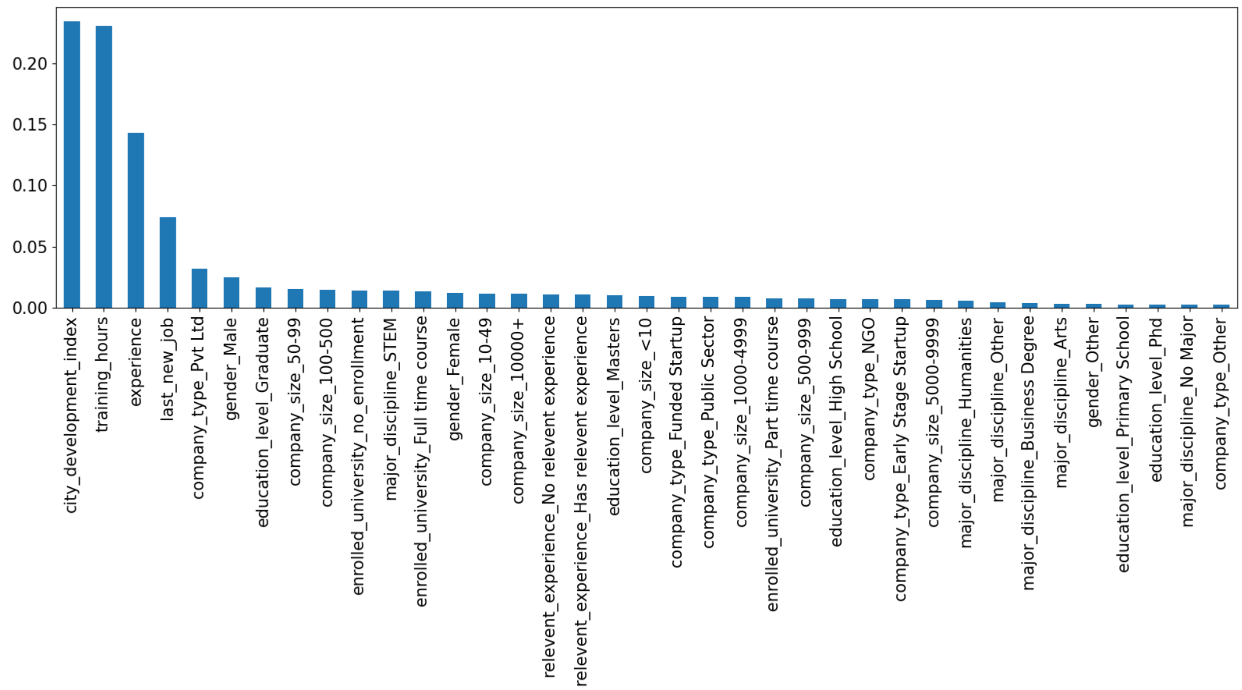
5)model evaluation

```
## Model evaluation
# report
print(classification_report(y_test,y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred) * 100)
print("ROC_AUC:", roc_auc_score(y_test,y_pred_score[:,-1]) * 100)

# confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
class_names = data["target"].unique()

df_cm = pd.DataFrame(conf_matrix, index=class_names, columns=class_names )
```

```
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
y_pred_score = clf.predict_proba(X_test)
fpr, tpr, _ = roc_curve(y_test, y_pred_score[:,-1])
auc = roc_auc_score(y_test, y_pred_score[:,-1])
#print(fpr)
#print(tpr)
#print(auc)
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()
```

6)Features Importance

```
# get feature importances
importances = clf.feature_importances_

# convert the importances into one-dimensional 1darray with corresponding df column names
as axis labels
f_importances = pd.Series(importances, X.columns)

# sort the array in descending order of the importances
f_importances.sort_values(ascending=False, inplace=True)

# make the bar Plot from f_importances
f_importances.plot(x='Features', y='Importance', kind='bar', figsize=(16, 9), rot=90,
fontsize=15)

# show the plot
plt.tight_layout()
plt.show()
```



7)final report EDA part and Appendix part.

Appendix

1. data head

2. data information

3. Null values

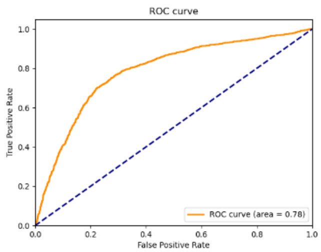
4. ROC_AUC for Logistic

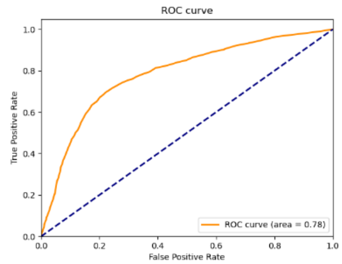
5. ROC_AUC for Random Forest

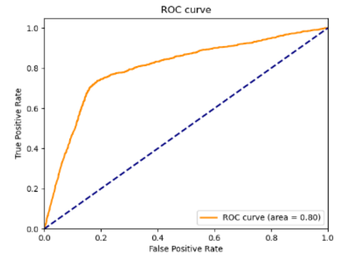
6.ROC_AUC for HGradient Boosting

```
Dataset first few rows:

   enrollee_id      city  ...  training_hours target
0         8949  city_103  ...              36    1.0
1        29725   city_40  ...              47    0.0
2        11561   city_21  ...              83    0.0
3        33241  city_115  ...              52    1.0
4          666  city_162  ...               8    0.0

[5 rows x 14 columns]
Dataset info:
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19158 entries, 0 to 19157
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   enrollee_id             19158 non-null  int64  
 1   city                    19158 non-null  object 
 2   city_development_index  19158 non-null  float64
 3   gender                  14650 non-null  object 
 4   relevent_experience     19158 non-null  object 
 5   enrolled_university     18772 non-null  object 
 6   education_level         18698 non-null  object 
 7   major_discipline        16345 non-null  object 
 8   experience              19093 non-null  object 
 9   company_size            13220 non-null  object 
 10  company_type            13018 non-null  object 
 11  last_new_job            18735 non-null  object 
 12  training_hours          19158 non-null  int64  
 13  target                  19158 non-null  float64
dtypes: float64(2), int64(2), object(10)
memory usage: 2.0+ MB
```







4. Results.

1)EDA

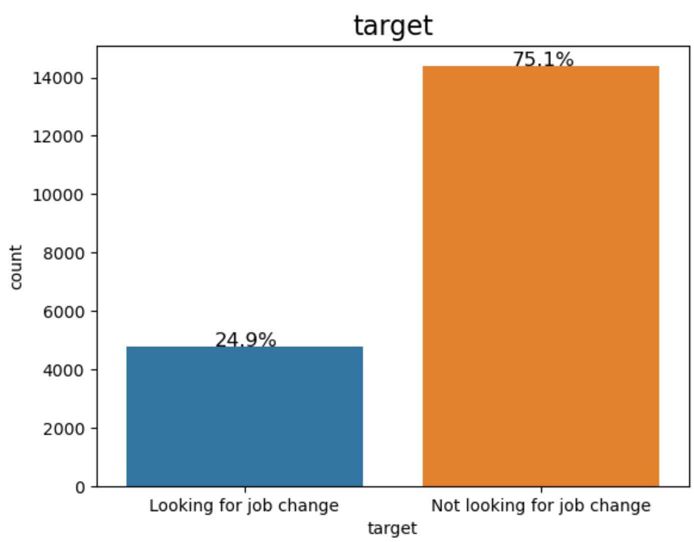


Figure.1 Counts and rates for people looking for job change

Over 19158 enrollees, 24.9% of them are looking for job change and 75.1% of them are not looking for a job (Figure.1).

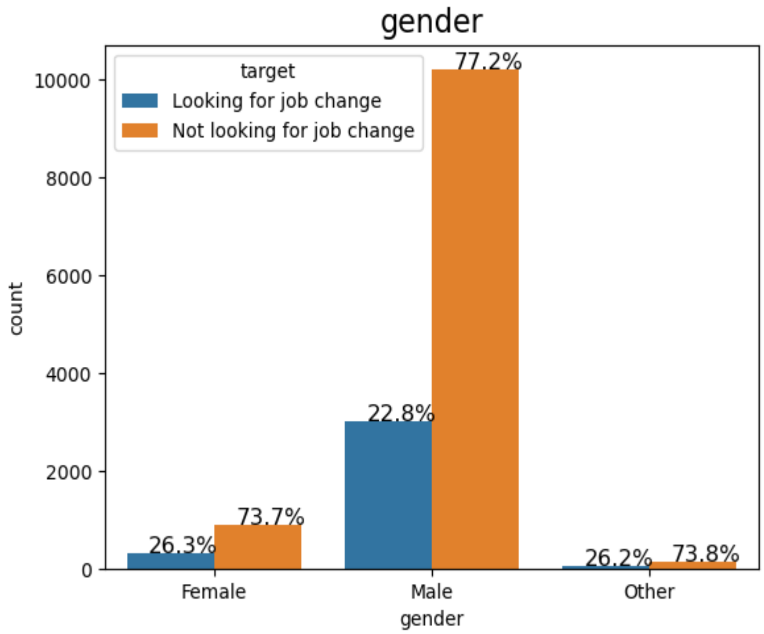


Figure.2 Distribution of job change by gender

With different genders，people shows a comparable rate of looking for a new job(Figure.2).

Figure.3 Distribution of job change by training_hours

People with different training hours show a comparable rate of looking for a new job(Figure.3).

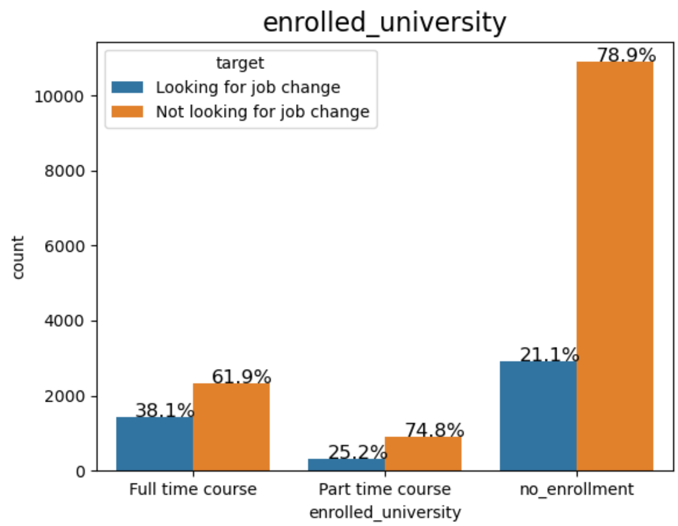


Figure 4. Distribution of job change by enrolled_university

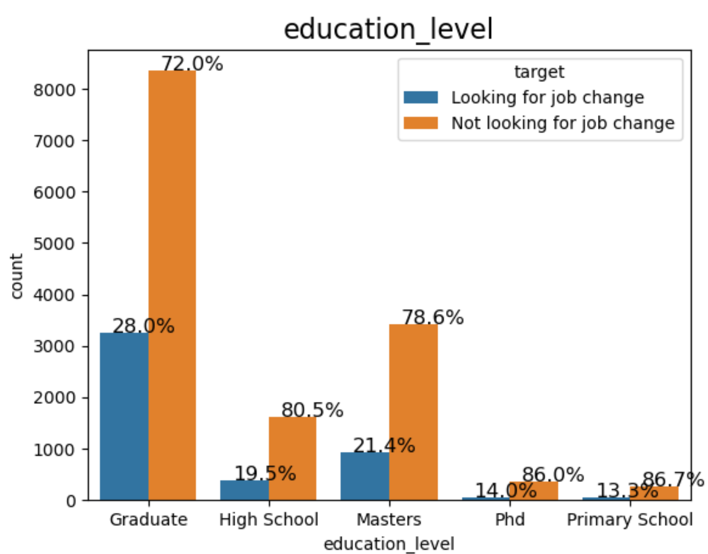


Figure 5. Distribution of job change by education_level

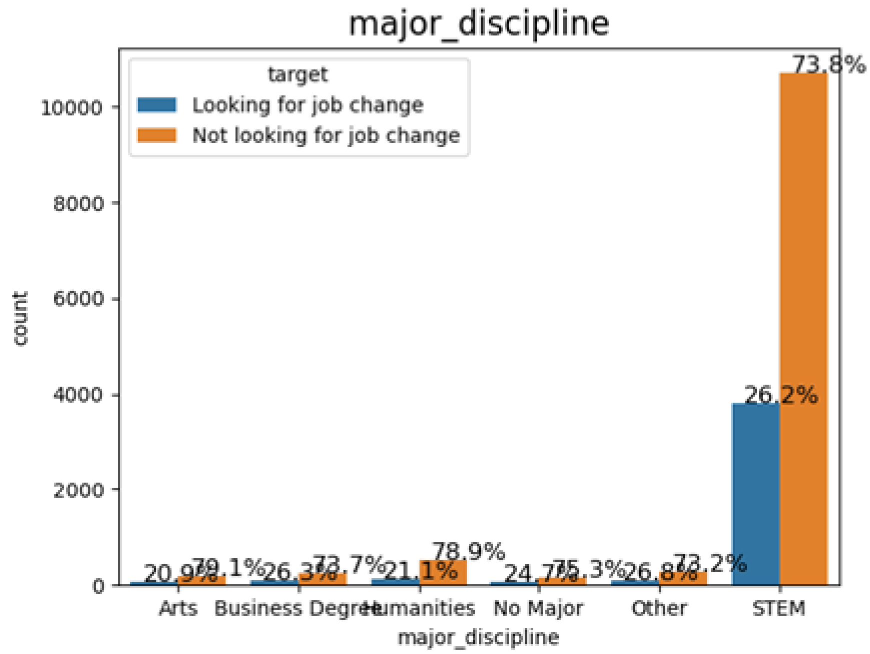


Figure 6. Distribution of job change by major_discipline

Education plays an important role in the rate of people looking for job change. People who took the full time course are more likely to look for a new job compared to others taking part-time course and no enrollment(Figure.4). People with graduate education level are more inclined to look for a new job compared to high school, masters, Ph.D and primary school(Figure.5). People with discipline of art and humanities are less likely to look for job change compared to the people with discipline of business, STEM and others(Figure.6).

Figure 7. Distribution of job change by city_development_index

City development index stand for the development level and stages, it is very interestingly that in the cities with lower city_development_index, the rate of people looking for a new job is significantly higher than that in the cities with higher city_development_index(Figure.7), which suggests that there are more opportunities in the development cities.

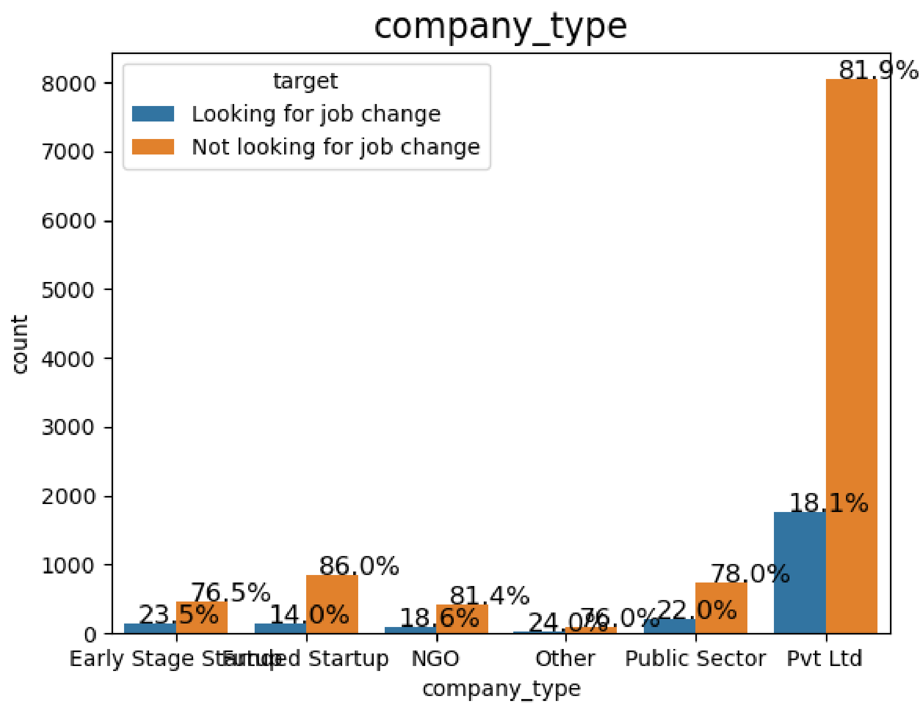


Figure.8 Distribution of job change by company_type

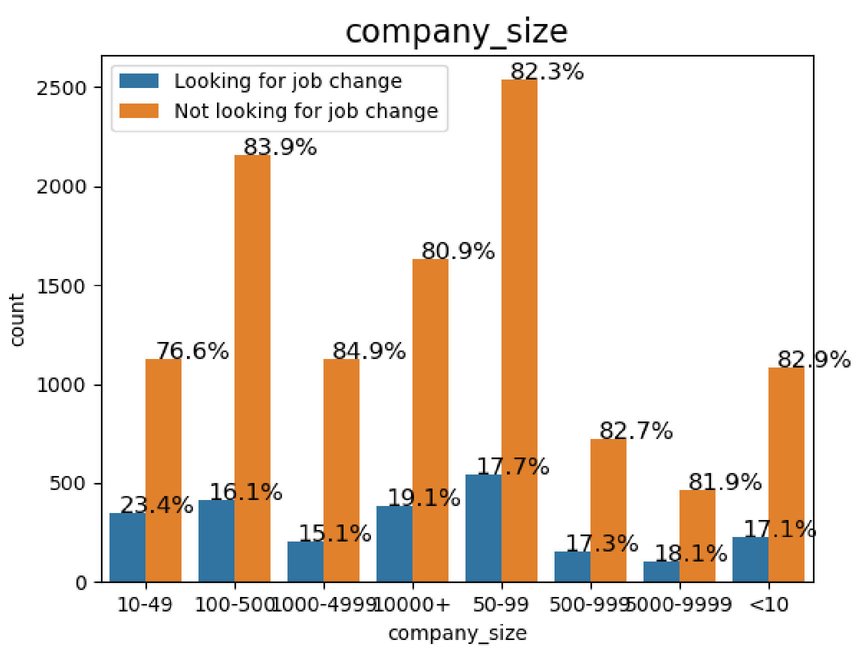


Figure.9 Distribution of job change by company_size

Company type and size also matters. People working in the Pvt Ltd, NGO and Founded Startup are less likely to look for a new job(Figure.8).People working in the company with size of 10-49 are more inclined to look for a new job(Figure.9).

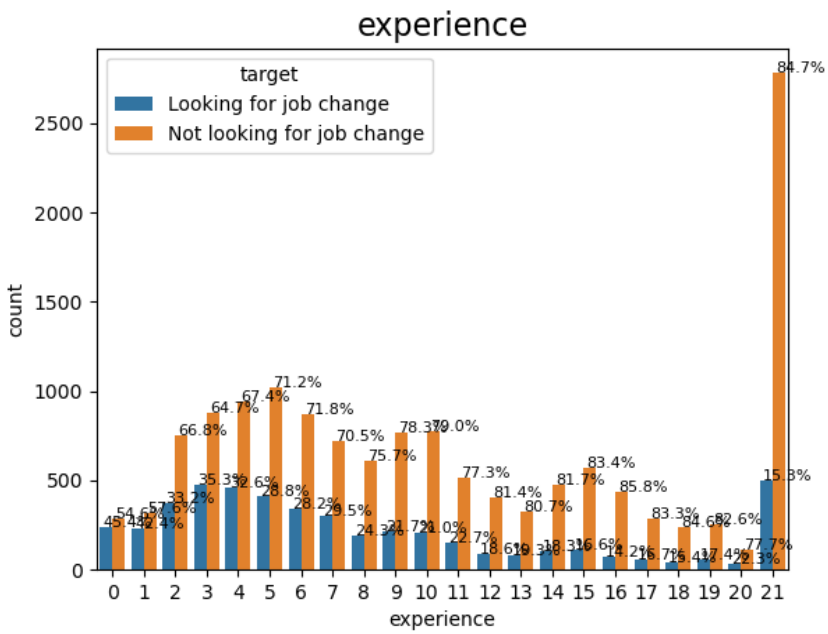


Figure.10 Distribution of job change by experience_years

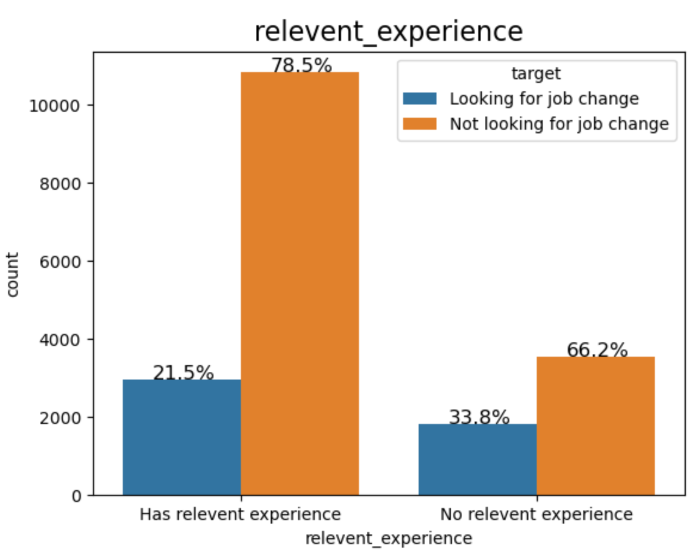


Figure.11 Distribution of job change by relevent_experience

Working experiences is an important factor affecting the rate of people looking for a new job. People with less working experiences are more likely to look for a new job(Figure.10). The rate of looking for a new job for people with no relevant experience is a little higher(Figure.11).

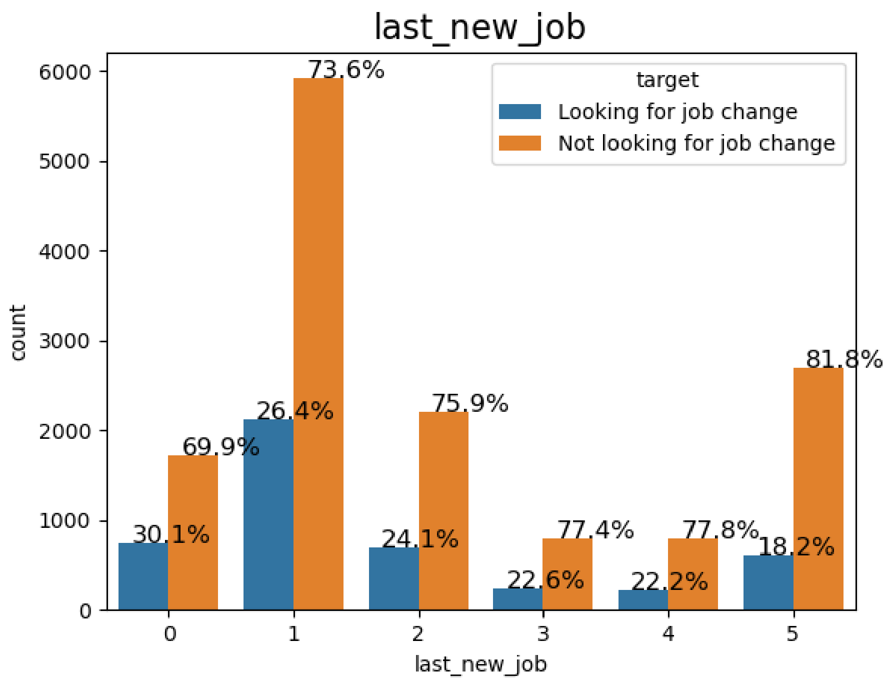
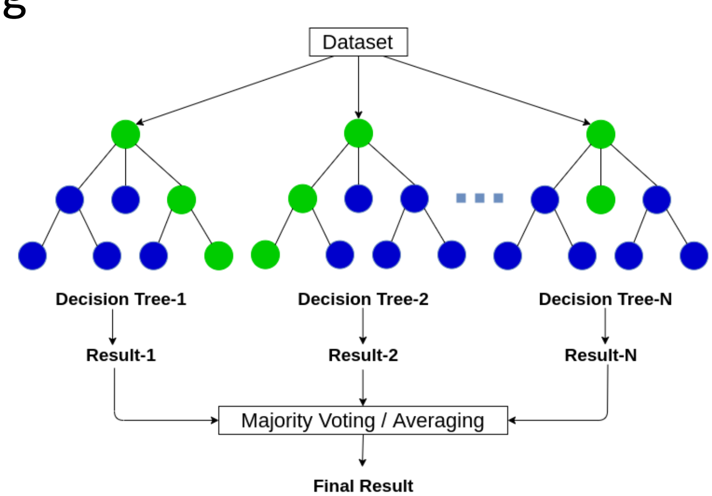


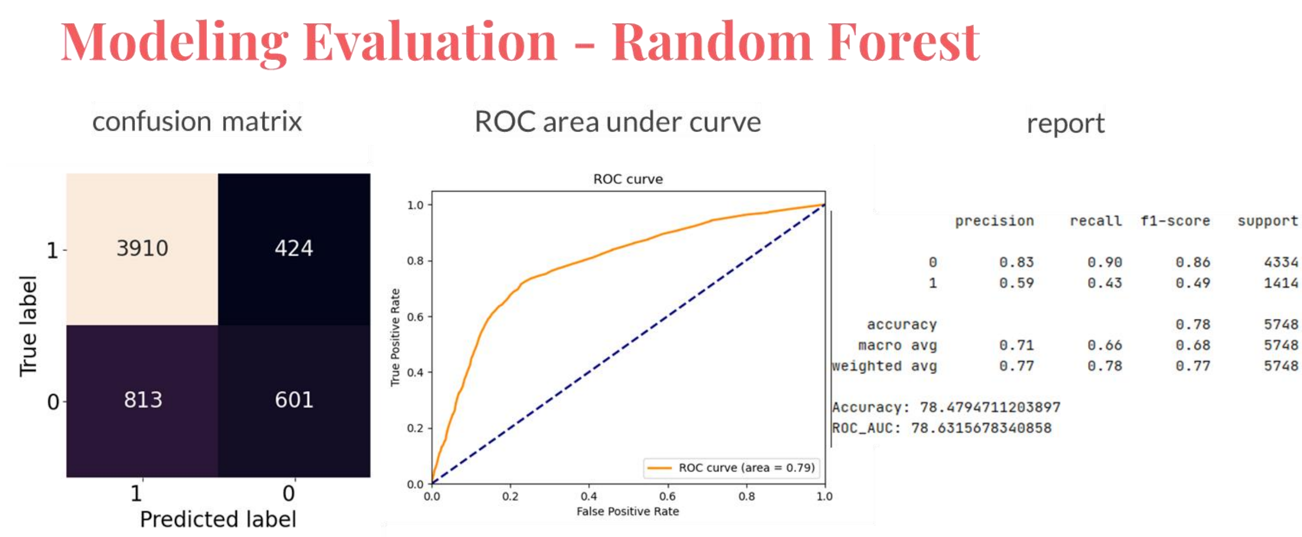
Figure.12 Distribution of job change by last_new_job

The difference of 1 year and zero year shows a significant higher rate of looking for a new job(Figure.12), which indicates that people looking for job change are used to working in different company for same time long.

2) Modeling



3)Model Evaluation



5. Summary and conclusions.

Based on the results for Random Forest model, KNN model, compared to the Logistic and Gradient Boosting models built by Renping, 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 and accuracy. Besides the models we talked about, I also conduct several other classifiers, like XGB boosting, CatBoosting, etc.… But none of them has the better result than the models we selected.

The GUI generation is not perfect in this project. I need to learn how to build a GUI and integrate all the data into it with PyQt5.

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

Base on the rules here,

the percentage =(620-133)/(620+201)=59.3%

7. References.

1) https://www.kaggle.com/arashnic/hr-analytics-job-change-of-data scientists?select=aug_train.csv
2) Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. Frontiers in neurorobotics, 7, 21.
3) 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.
4) Menard, S. (2002). Applied logistic regression analysis (Vol. 106)
5) https://christophm.github.io/interpretable-ml-book/logistic.html
6) https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
7) https://www.javatpoint.com/classification-algorithm-in-machine-learning