SAMSUNG

Together for Tomorrow!
Enabling People
Education for Future Generations

Samsung Innovation Campus

Artificial Intelligence Course

HR Analysis

Machine Learning project

AI Course

Samsung Innovation Campus

Hunters Team
Group:

Ola El-Shiekh

https://www.linkedin.com/in/ola-el-shiekh/

Ahmed Abdelnasser

https://www.linkedin.com/in/ahmed-abdelnasser-sayed

Presentation Contents

slide	content
1	• Why data analysis ?
2:3	• Our Data : HR Analysis
4:5	• Data insights before working on it:
6:14	• Data insights after working on it:
14:27	• modeling
28	• Finishing

Why Data Analytics?

Data analytics is important because it helps businesses optimize their performances. Implementing it into the business model means companies can help them in many decisions

Also **Data visualizations** make big and small data easier for the human to understand, and visualization also makes it easier to detect patterns, trends, and outliers in groups of data.

Our Data : HR Analysis

Our data is about Company gets large number of signups for their trainings. Now, company wants to connect these enrollees with their clients who are looking to hire employees working in the same domain. Before that, it is important to know which of these candidates are really looking for a new employment

the we worked to design a model that uses the current credentials/demographics/experience to predict the probability of an enrollee to look for a new job.

Our data : HR Analysis

Data contains ：

enrollee_id: which is special number for each enrolee

City ： city where he lives

city_development_index

gender ： male, female or other

relevent_experience : having previous experience related to job

enrolled_university : if he studing in uni or not

education_level : his education level when he enroll

major_discipline : his major of study and work

Experience : num. of years of experience

company_size : number of employees in company

company_type ： company type

last_new_job: last new job

training_hours: number of training hours

Data insights before working on it:

```
train.isna().sum()
```

```
enrollee_id                0
city                       0
city_development_index     0
gender                  4098
relevent_experience        0
enrolled_university      342
education_level          457
major_discipline        2838
experience                59
company_size            4779
company_type            5039
last_new_job             367
training_hours             0
target                     0
dtype: int64
```

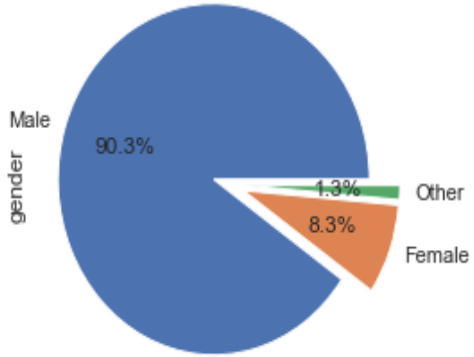
```
test.isna().sum()
```

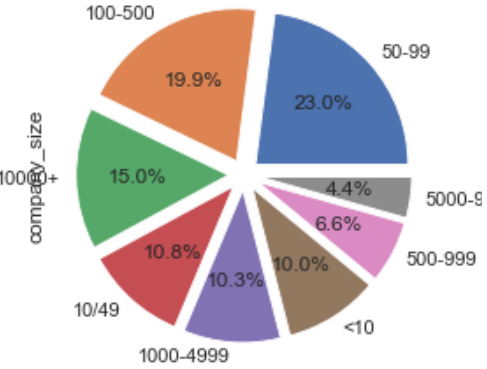
```
enrollee_id                0
city                       0
city_development_index     0
gender                  3388
relevent_experience        0
enrolled_university      279
education_level          395
major_discipline        2393
experience                44
company_size            4051
company_type            4330
last_new_job             304
training_hours             0
dtype: int64
```

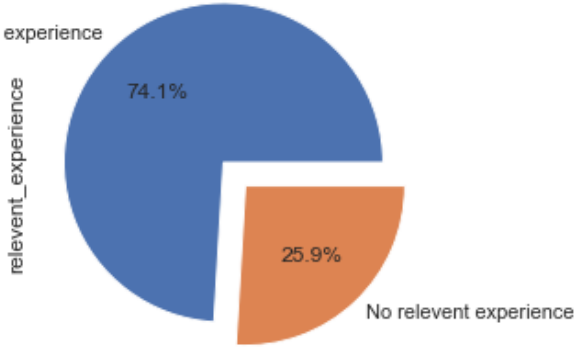
We have a lot of missing data that we caused a problems so we need to handle this.

Data insights before working on it:

- **We have to keep percentages as we can to avoid errors :**

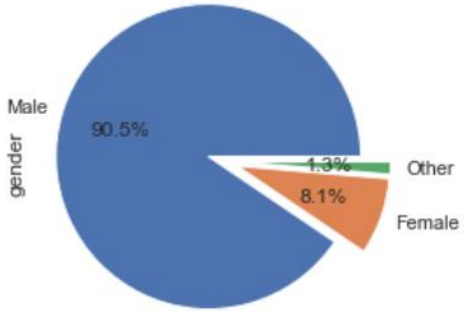


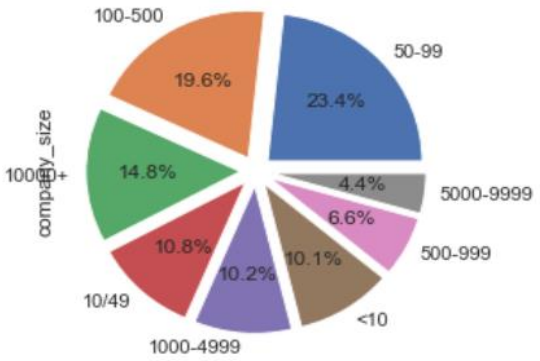


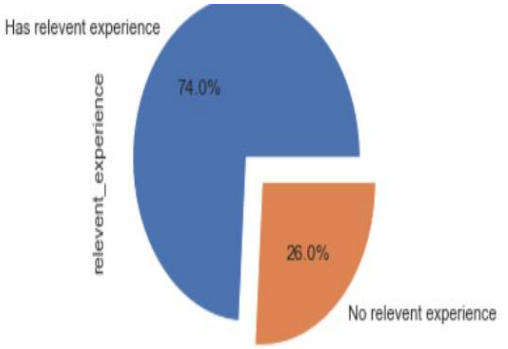


Data insights after working on it:

We use Filling values forward technique to fill our missing data and that give us good values and close to values before filling :







Data insights after working on it:

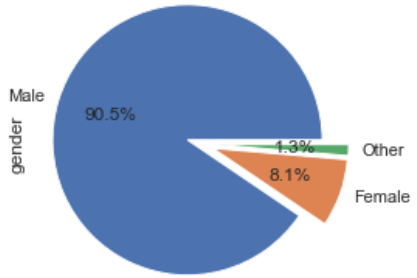
And we have no missing data!!

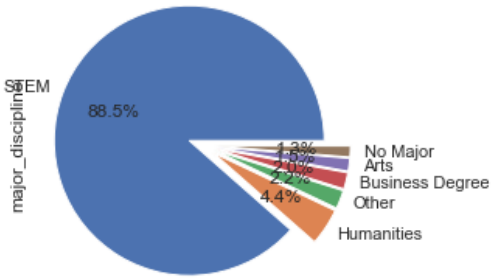
```
train.isnull().sum()
```

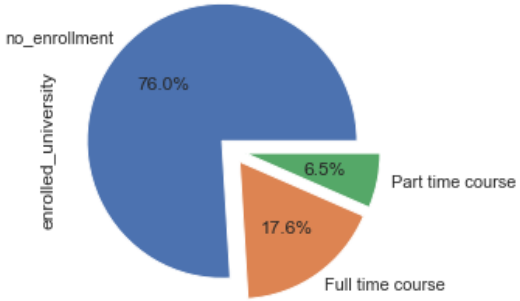
```
enrollee_id             0
city                    0
city_development_index  0
gender                  0
relevent_experience     0
enrolled_university     0
education_level         0
major_discipline        0
experience              0
company_size            0
company_type            0
last_new_job            0
training_hours          0
target                  0
dtype: int64
```

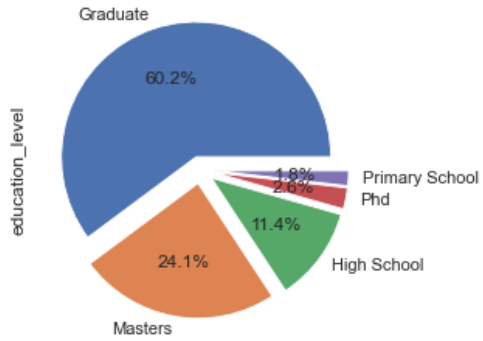
Data insights after working on it:

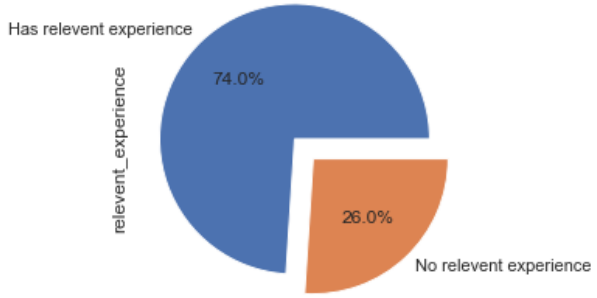
Now, let's show up data with visuals to know more about it…

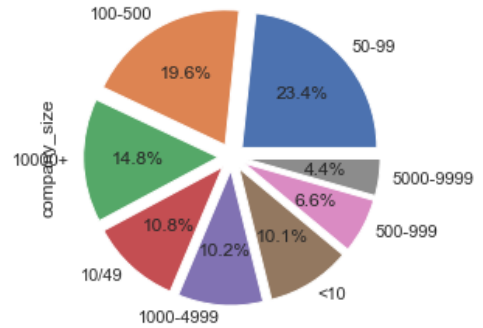






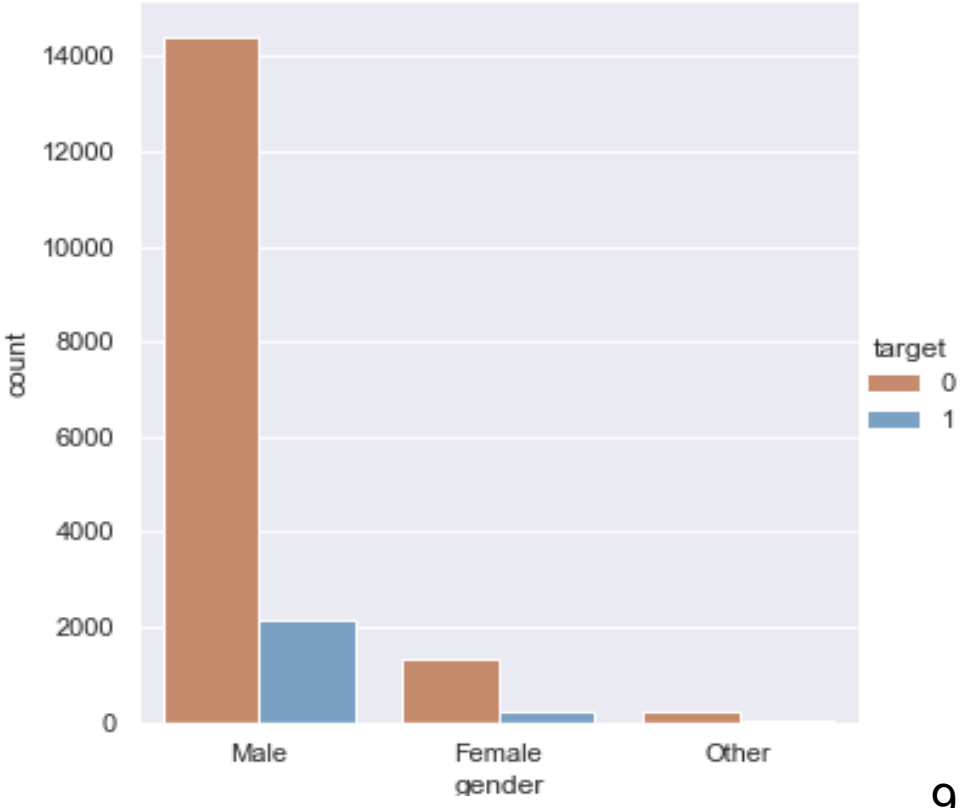


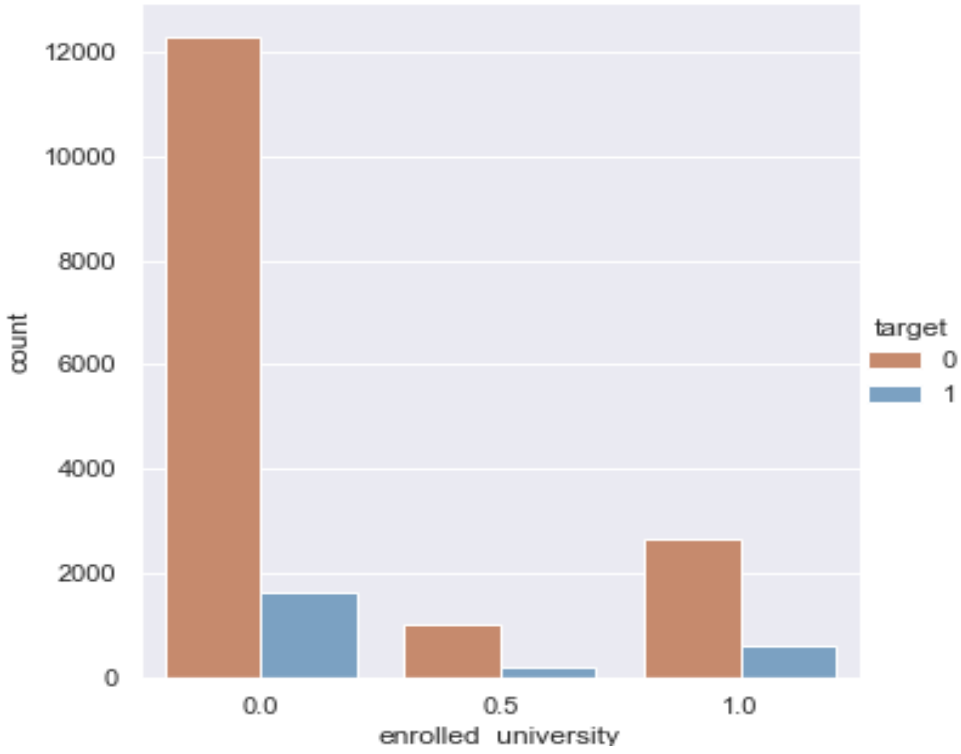




Data insights after working on it:

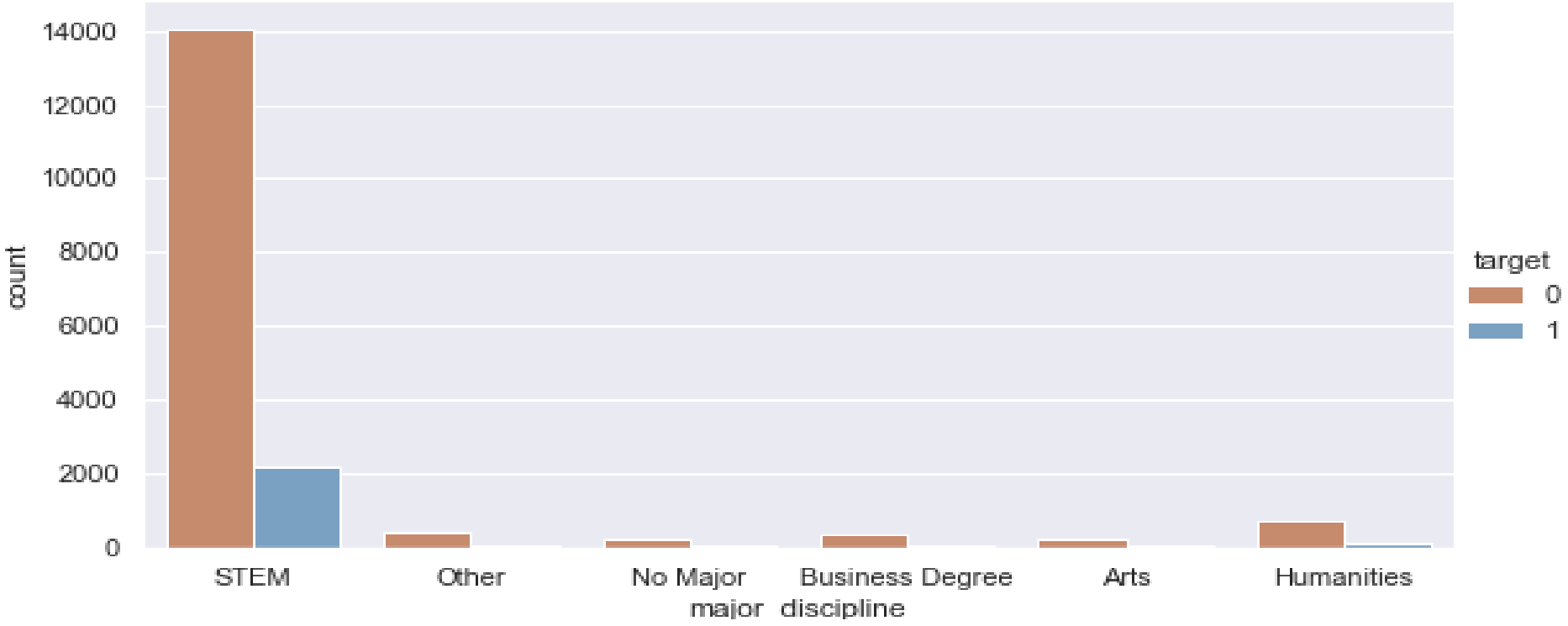
And here we show up charts clarify ratio and numbers between features and target in train data :



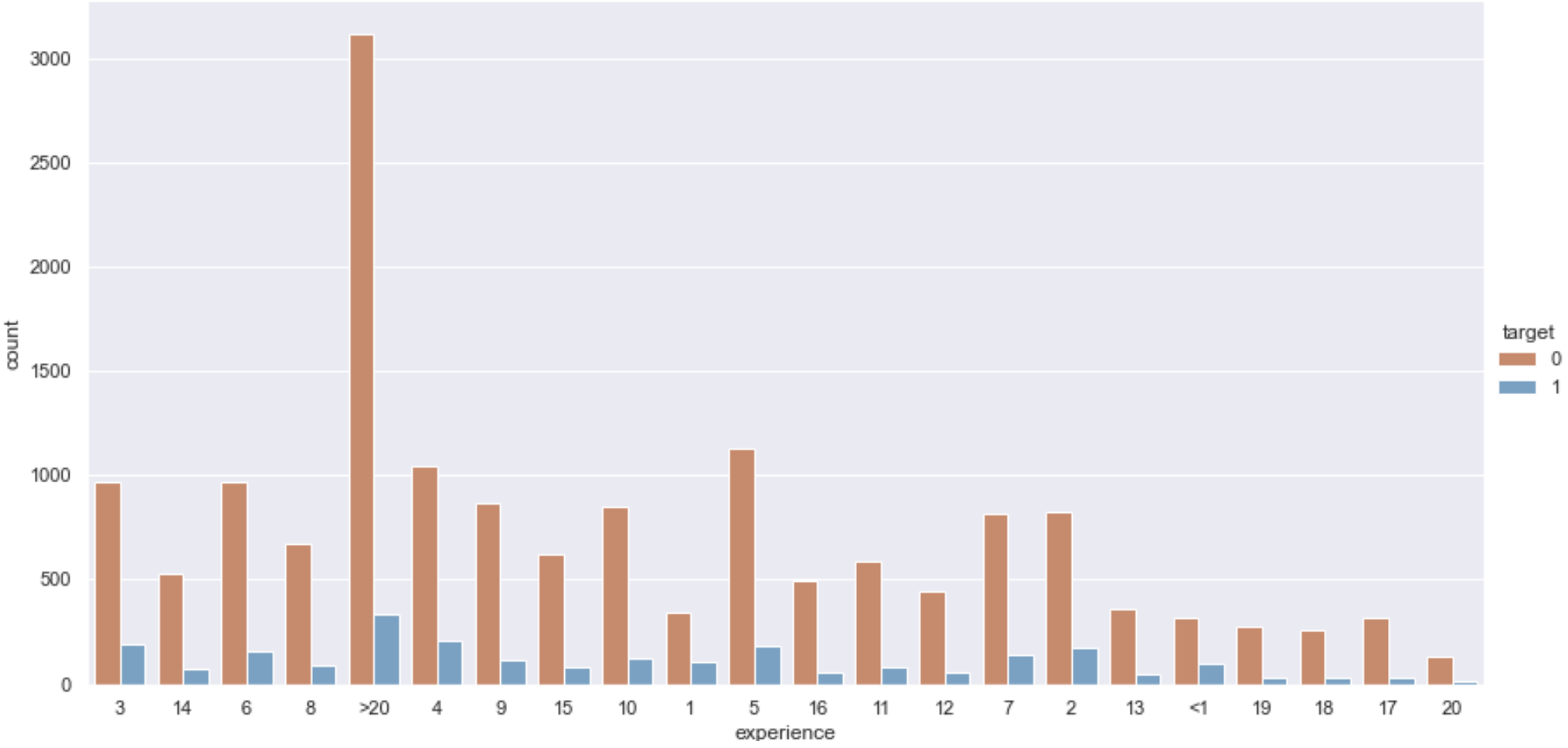


Data insights after working on it:

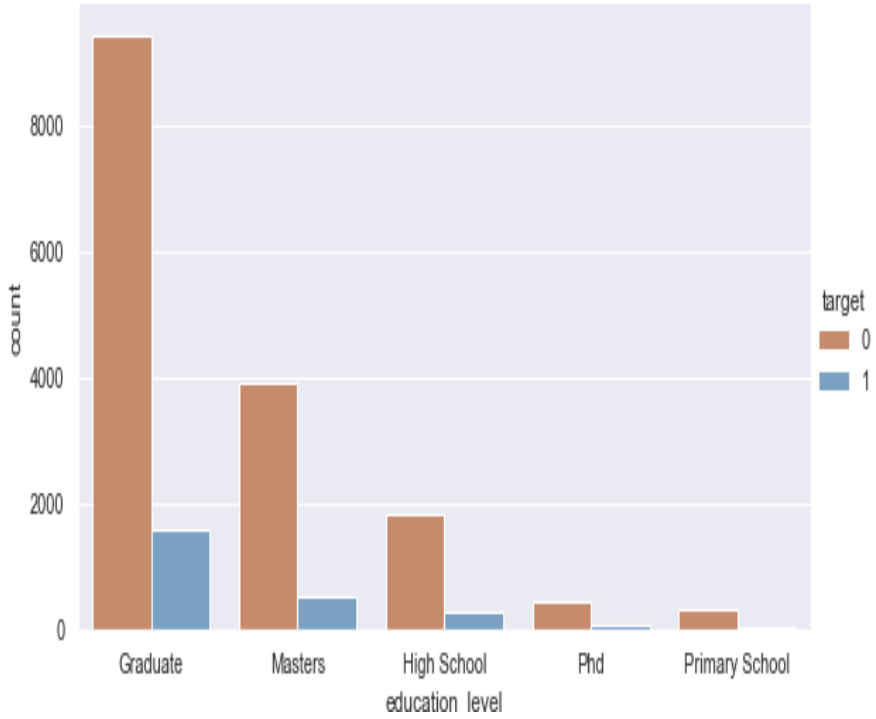
And here we show up charts clarify ratio and numbers between features and target in train data :

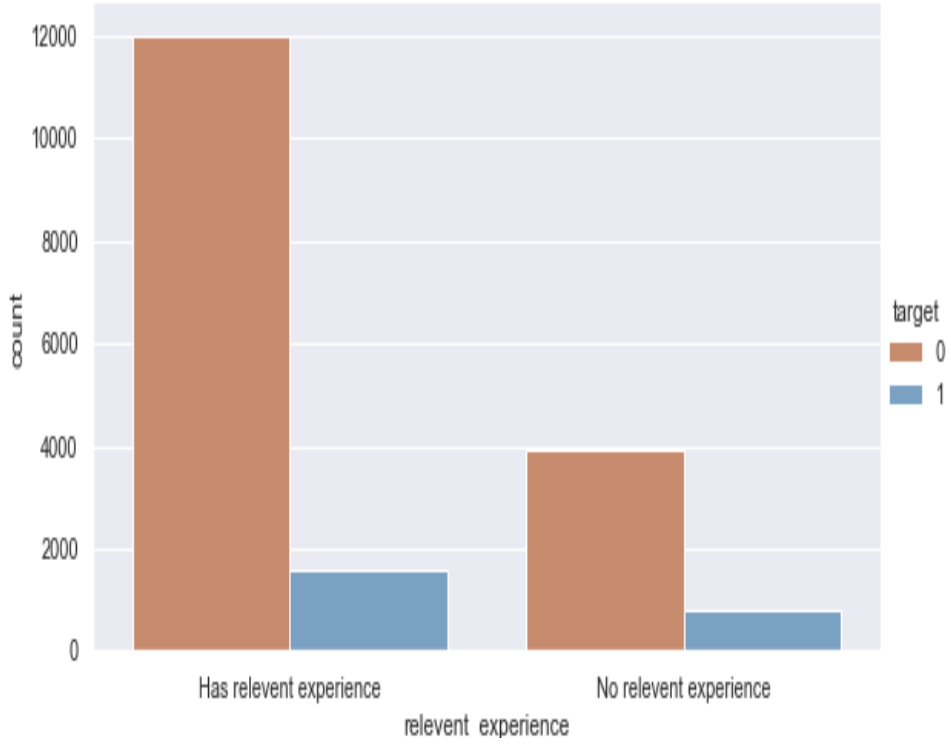


Data insights after working on it:



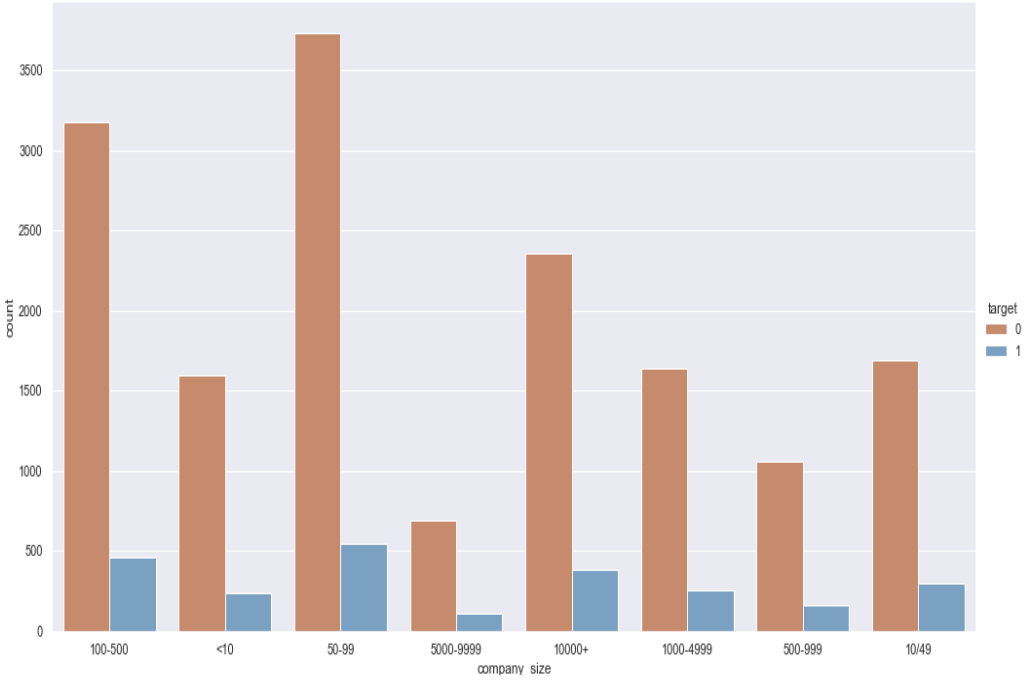
Data insights after working on it:

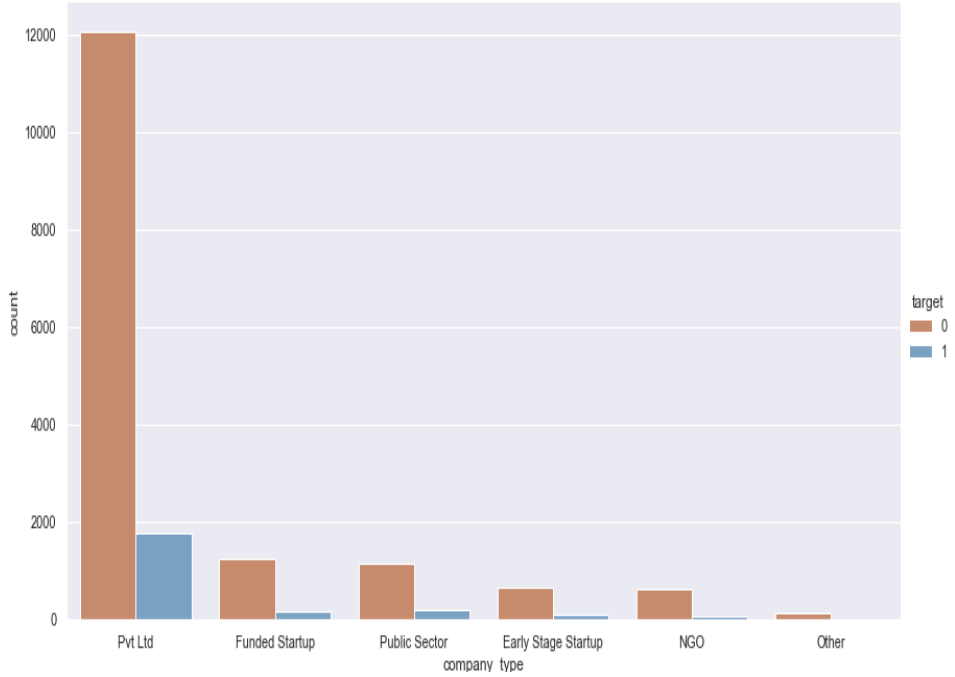




12

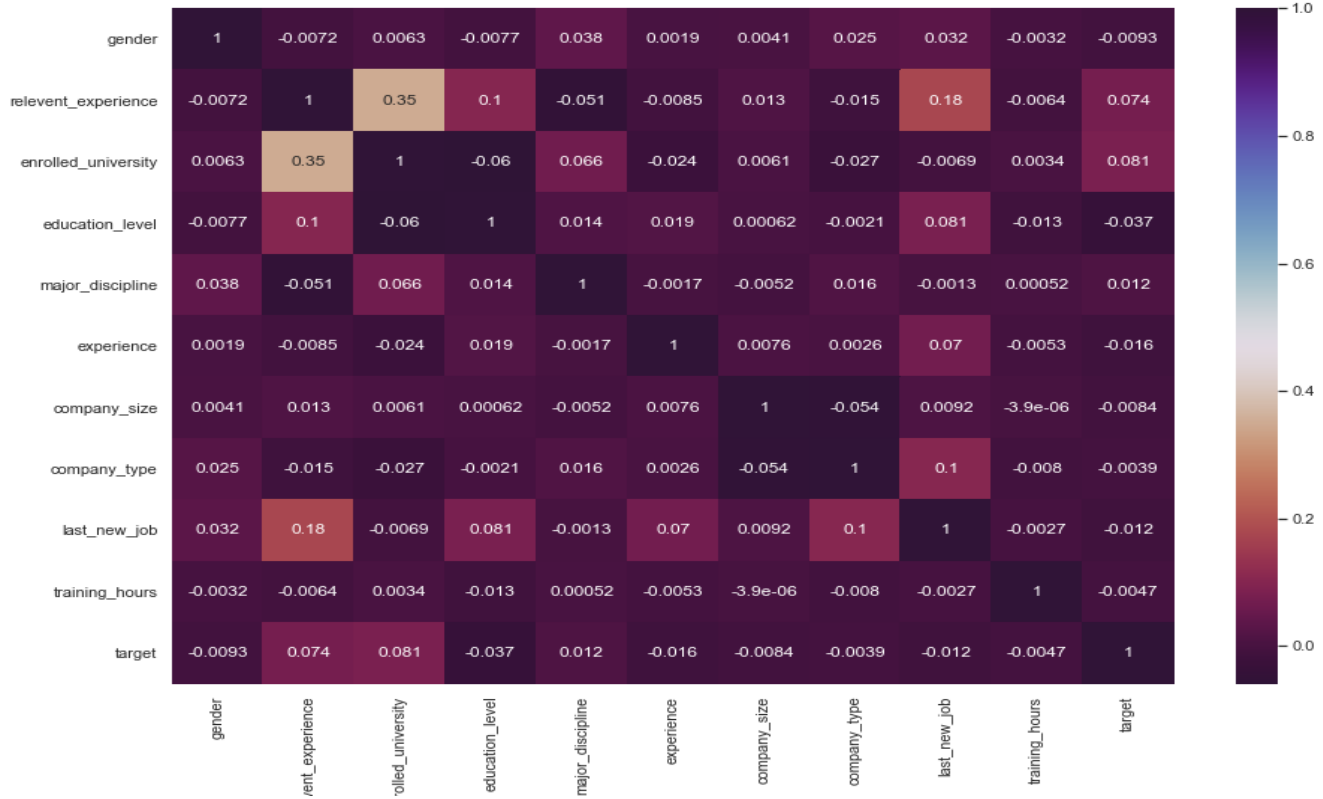
Data insights after working on it:





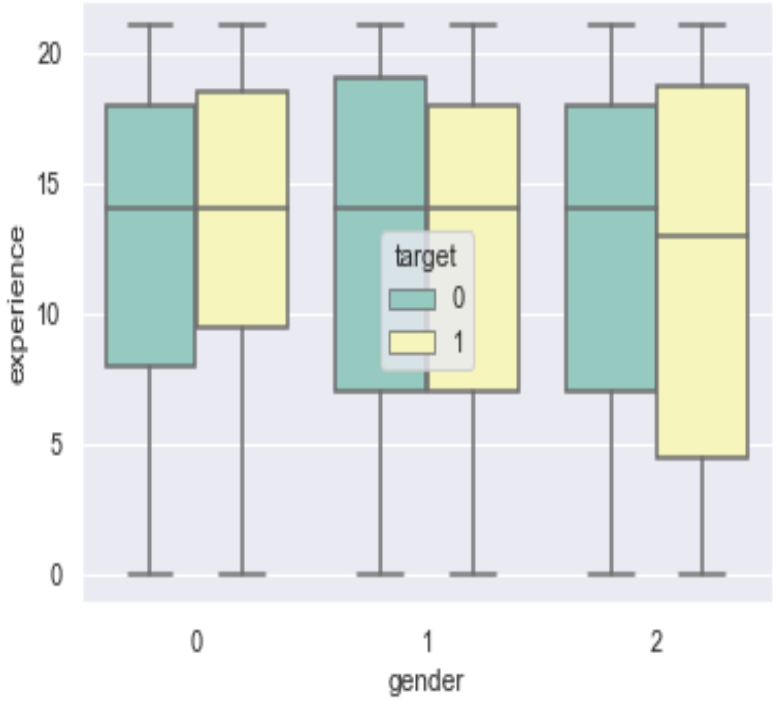
Data insights after working on it:

After encoding data we can see relation between any two features below by using correlation heat map:



Data insights after working on it:

Here we use boxplot to show up target statistics according to gender and experience :



Conclusion

The most qualified applicant for the data analysis Internship :

•Gender : Male (There is gender equality despite of the number of male applicants is more than females but the targeting is nearly equivalent). This is shown in the boxplot above.

•More than 20 years experience.

•Has relevant experience.

•Last job was at PVT LTD Company.

•Not currently enrolled at University.

•Major is STEM.

•Graduated

Modeling :

We working on different models to detect suitable one and we get :

Logistic Regression

```
# Confusion matrix.
conf_mat = metrics.confusion_matrix(Y_test,Y_pred_test)
print(conf_mat)
```

```
[[6346    0]
 [ 998    0]]
```

```
# Alternative way.
accuracy = metrics.accuracy_score(Y_test,Y_pred_test)
Recall = metrics.recall_score(Y_test,Y_pred_test)
precision = metrics.precision_score(Y_test,Y_pred_test)
print('Accuracy    = {}'.format(np.round(accuracy,3)))
print('Recall  = {}'.format(np.round(Recall,3)))
print('Precision   = {}'.format(np.round(precision,3)))
```

```
Accuracy    = 0.864
Recall  = 0.0
Precision   = 0.0
```

Modeling :

KNN

```
kclf.score(X_train,Y_train)
```

```
0.876078075351793
```

```
kclf.score(X_test,Y_test)
```

```
0.8523965141612201
```

```
confusion_matrix(Y_test,kclf.predict(X_test))
```

```
array([[6222,  124],
       [ 960,   38]], dtype=int64)
```

```
# Alternative way.
accuracy = metrics.accuracy_score(Y_test,Y_pred_test)
Recall = metrics.recall_score(Y_test,Y_pred_test)
precision = metrics.precision_score(Y_test,Y_pred_test)
print('Accuracy    = {}'.format(np.round(accuracy,3)))
print('Recall  = {}'.format(np.round(Recall,3)))
print('Precision   = {}'.format(np.round(precision,3)))
```

```
Accuracy    = 0.852
Recall  = 0.038
Precision   = 0.235
```

Modeling :

SVM

```
svm.score(X_test,Y_test)
```

```
0.8641067538126361
```

```
svm.score(X_train,Y_train)
```

```
0.8704493871992737
```

```
confusion_matrix(Y_test,svm.predict(X_test))
```

```
array([[6346,    0],
       [ 998,    0]], dtype=int64)
```

```
# Alternative way.
accuracy = metrics.accuracy_score(Y_test,Y_pred_test)
Recall = metrics.recall_score(Y_test,Y_pred_test)
precision = metrics.precision_score(Y_test,Y_pred_test)
print('Accuracy     = {}'.format(np.round(accuracy,3)))
print('Recall   = {}'.format(np.round(Recall,3)))
print('Precision    = {}'.format(np.round(precision,3)))
```

```
Accuracy     = 0.864
Recall   = 0.0
Precision    = 0.0
```

Modeling :

Navie Bayes

```
gnb.score(X_train,Y_train)
```

```
0.8651838402178847
```

```
gnb.score(X_test,Y_test)
```

```
0.858660130718954 2
```

```
confusion_matrix(Y_test,gnb.predict(X_test))
```

```
array([[6294,   52],
       [ 986,   12]], dtype=int64)
```

```
# Alternative way.
accuracy = metrics.accuracy_score(Y_test,Y_pred_test)
Recall = metrics.recall_score(Y_test,Y_pred_test)
precision = metrics.precision_score(Y_test,Y_pred_test)
print('Accuracy    = {}'.format(np.round(accuracy,3)))
print('Recall  = {}'.format(np.round(Recall,3)))
print('Precision   = {}'.format(np.round(precision,3)))
```

```
Accuracy    = 0.859
Recall  = 0.012
Precision   = 0.188
```

Modeling :

Random Forest

```
confusion_matrix(Y_test,clf.predict(X_test))
```

```
array([[6250,   96],
       [ 973,   25]], dtype=int64)
```

```
print("random forest F1-score",f1_score(Y_test,Y_pred_test))
print("random forest Recall: ",recall_score(Y_test,Y_pred_test))
```

```
random forest F1-score 0.044682752457551385
random forest Recall:  0.025050100200400802
```

```
# Alternative way.
accuracy = metrics.accuracy_score(Y_test,Y_pred_test)
Recall = metrics.recall_score(Y_test,Y_pred_test)
precision = metrics.precision_score(Y_test,Y_pred_test)
print('Accuracy    = {}'.format(np.round(accuracy,3)))
print('Recall  = {}'.format(np.round(Recall,3)))
print('Precision   = {}'.format(np.round(precision,3)))
```

```
Accuracy    = 0.854
Recall  = 0.025
Precision   = 0.207
```

Modeling :

Logistic Regression with grid search

```
confusion_matrix(Y_test,clf2_)
```

```
Out[94]: array([[6346,    0],
                [ 998,    0]], dtype=int64)
```

Modeling :

XGB Model

```
xgb.score(X_train,Y_train)
```

```
0.9769405356332275
```

```
xgb.score(X_test,Y_test)
```

```
0.8458605664488017
```

```
xgb.score(X_train,Y_train)
```

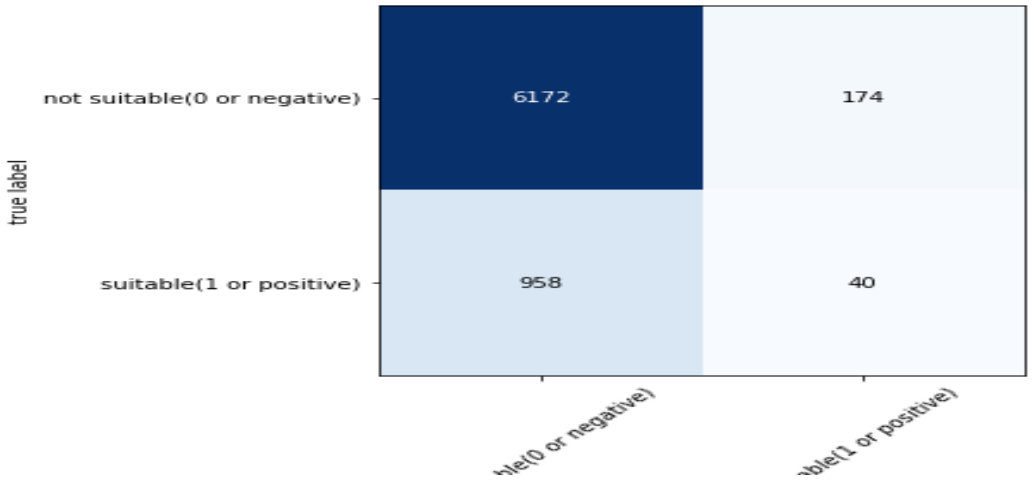
```
0.9769405356332275
```

```
xgb.score(X_test,Y_test)
```

```
0.8458605664488017
```

```
conf_mat=confusion_matrix(Y_test,xgb.predict(X_test))
```

```
plot_confusion_matrix(conf_mat,class_names=["not suitable(0 or nega
```



Modeling :

Continue XGB

In [192]:
```
# Alternative way.
accuracy = metrics.accuracy_score(Y_test,Y_pred_test)                    # Alternative way to calculate the accuracy.
Recall = metrics.recall_score(Y_test,Y_pred_test)
precision = metrics.precision_score(Y_test,Y_pred_test)
print('Accuracy    = {}'.format(np.round(accuracy,3)))
print('Recall  = {}'.format(np.round(Recall,3)))
print('Precision   = {}'.format(np.round(precision,3)))
```

```
Accuracy    = 0.846
Recall  = 0.04
Precision   = 0.187
```

Modeling :

SO, After Modelling I See that the XGBoost classifier gets the best results.

Modeling :

Finally, we predict target and fill sample with it.

```
In [479]: pred_xg = Xg_boost(X_train,Y_train,X_test)
```

```
[18:15:22] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:573:
Parameters: { "criterion", "loss", "min_impurity_decrease", "min_samples_leaf", "min_samples_split", "min_weight_fraction_lea
f", "presort", "tol", "validation_fraction", "verbose", "warm_start" } might not be used.

  This may not be accurate due to some parameters are only used in language bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through this
  verification. Please open an issue if you find above cases.


[18:15:22] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly se
t eval_metric if you'd like to restore the old behavior.
```

```
In [480]: sample['target'] = pred_xg
print(sample['target'].unique())
sample.to_csv('XG.csv',index = False)
```

```
[1 0]
```

```
In [481]: dict(sample['target'])
```

```
 272: 0,
 273: 1,
 274: 0,
 275: 0,
 276: 0,
 277: 0,
 278: 0,
 279: 0,
 280: 0,
 281: 0,
 282: 0,
 283: 0,
 284: 1,
 285: 0,
 286: 0,
 287: 0,
 288: 0,
 289: 0,
 290: 0,
```

Code on Kaggle and github.

Ola's Account:

https://www.kaggle.com/olaelshiekh/first-code-with-ahmed

https://github.com/olaelshiekh/HR-Analysis-Preprocessing-Modelling-

Ahmed's Account:

https://www.kaggle.com/ahmedabdelnasser1/first-code-with-ola

https://github.com/ahmed3bnaser/HR-Analysis-Project

Special Thanks for :

Dr: Doaa Mahmoud

Eng : Shimaa Osman

THANKS!!


THANK YOU FOR LISTENING
TO MY PRESENTATION
memegenerator.net

SAMSUNG

Together for Tomorrow!

Enabling People

Education for Future Generations