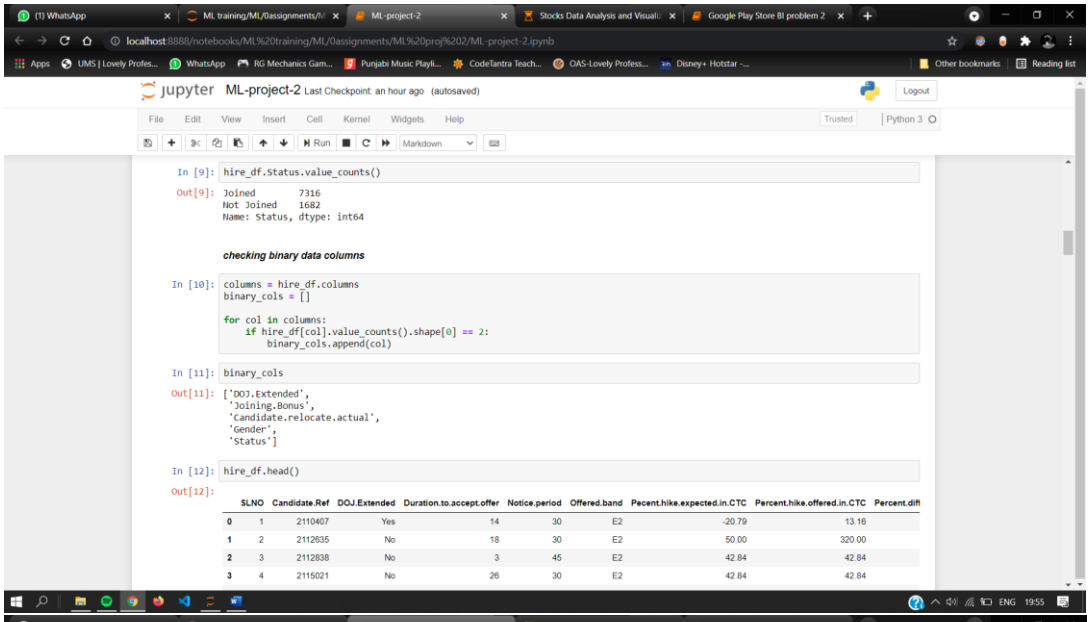


jupyter ML-project-2 Last Checkpoint: an hour ago (autosaved)

In [9]:
```
hire_df.Status.value_counts()
```

Out[9]:
```
Joined        7316
Not Joined    1682
Name: Status, dtype: int64
```

checking binary data columns

In [10]:
```
columns = hire_df.columns
binary_cols = []

for col in columns:
    if hire_df[col].value_counts().shape[0] == 2:
        binary_cols.append(col)
```

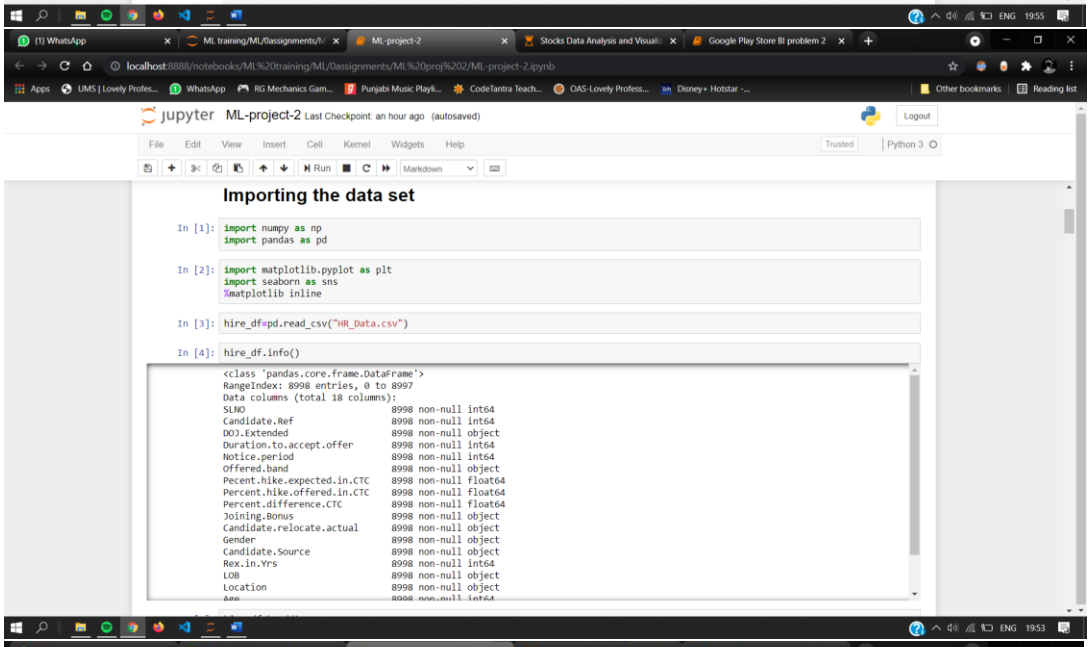
In [11]:
```
binary_cols
```

Out[11]:
```
['DOJ.Extended',
 'Joining.Bonus',
 'Candidate.relocate.actual',
 'Gender',
 'Status']
```

In [12]:
```
hire_df.head()
```

Out[12]:

	SLNO	Candidate.Ref	DOJ.Extended	Duration.to.accept.offer	Notice.period	Offered.band	Pecent.hike.expected.in.CTC	Percent.hike.offered.in.CTC	Percent.diff
0	1	2110407	Yes	14	30	E2	-20.79	13.16	
1	2	2112635	No	18	30	E2	50.00	320.00	
2	3	2112838	No	3	45	E2	42.84	42.84	
3	4	2115021	No	26	30	E2	42.84	42.84	



jupyter ML-project-2 Last Checkpoint: an hour ago (autosaved)

Importing the data set

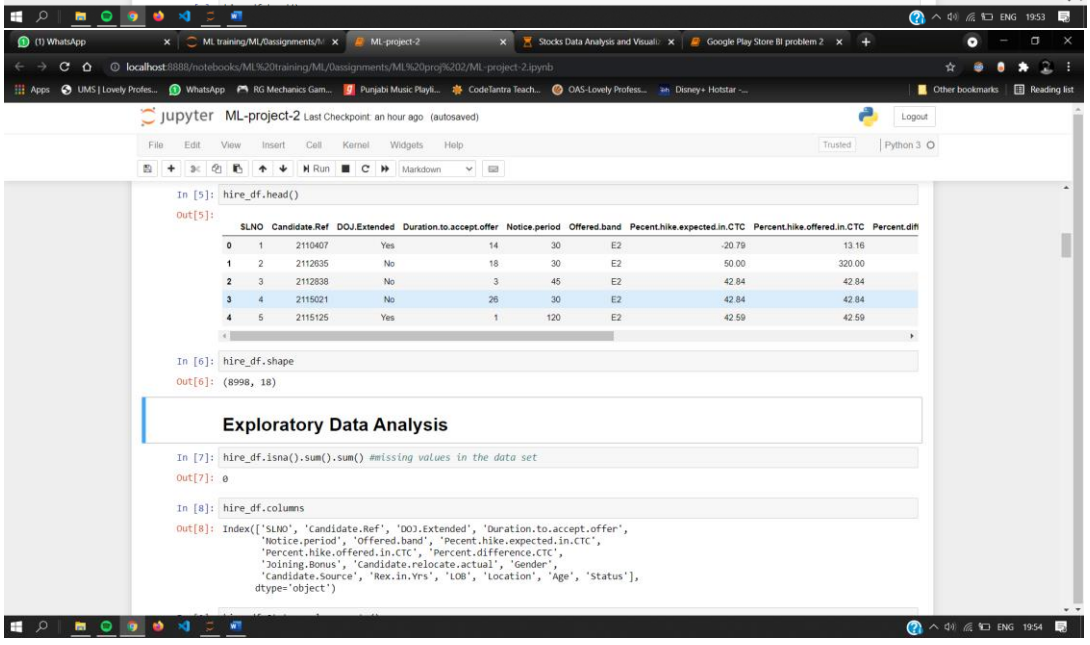
In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:
```
hire_df=pd.read_csv("HR_Data.csv")
```

In [4]:
```
hire_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8998 entries, 0 to 8997
Data columns (total 18 columns):
SLNO                        8998 non-null int64
Candidate.Ref               8998 non-null int64
DOJ.Extended                8998 non-null object
Duration.to.accept.offer    8998 non-null int64
Notice.period               8998 non-null int64
Offered.band                8998 non-null object
Pecent.hike.expected.in.CTC 8998 non-null float64
Percent.hike.offered.in.CTC 8998 non-null float64
Percent.difference.CTC      8998 non-null float64
Joining.Bonus               8998 non-null object
Candidate.relocate.actual   8998 non-null object
Gender                      8998 non-null object
Candidate.Source            8998 non-null object
Rex.in.Yrs                  8998 non-null int64
LOB                         8998 non-null object
Location                    8998 non-null object
Age                         8998 non-null int64
```



jupyter ML-project-2 Last Checkpoint: an hour ago (autosaved)

In [5]:
```
hire_df.head()
```

Out[5]:

	SLNO	Candidate.Ref	DOJ.Extended	Duration.to.accept.offer	Notice.period	Offered.band	Pecent.hike.expected.in.CTC	Percent.hike.offered.in.CTC	Percent.diff
0	1	2110407	Yes	14	30	E2	-20.79	13.16	
1	2	2112635	No	18	30	E2	50.00	320.00	
2	3	2112838	No	3	45	E2	42.84	42.84	
3	4	2115021	No	26	30	E2	42.84	42.84	
4	5	2115125	Yes	1	120	E2	42.59	42.59	

In [6]:
```
hire_df.shape
```

Out[6]:
```
(8998, 18)
```

Exploratory Data Analysis

In [7]:
```
hire_df.isna().sum().sum() #missing values in the data set
```

Out[7]:
```
0
```

In [8]:
```
hire_df.columns
```

Out[8]:
```
Index(['SLNO', 'Candidate.Ref', 'DOJ.Extended', 'Duration.to.accept.offer',
       'Notice.period', 'Offered.band', 'Pecent.hike.expected.in.CTC',
       'Percent.hike.offered.in.CTC', 'Percent.difference.CTC',
       'Joining.Bonus', 'Candidate.relocate.actual', 'Gender',
       'Candidate.Source', 'Rex.in.Yrs', 'LOB', 'Location', 'Age', 'Status'],
      dtype='object')
```

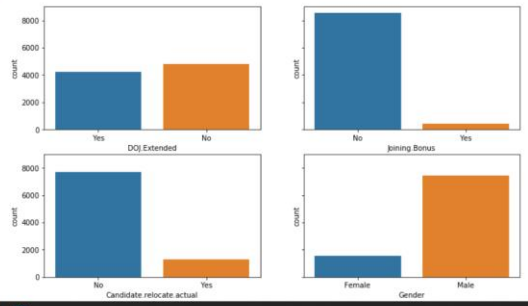
jupyter ML-project-2 Last Checkpoint: an hour ago (autosaved)

In [13]:
```
#class distribution of binary features.

fig, axes = plt.subplots(2, 2, figsize=(12, 7), sharey=True)

sns.countplot("DOJ.Extended", data=hire_df, ax=axes[0,0])
sns.countplot("Joining.Bonus", data=hire_df, ax=axes[0,1])
sns.countplot("Candidate.relocate.actual", data=hire_df, ax=axes[1,0])
sns.countplot("Gender", data=hire_df, ax=axes[1,1])
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x2028492b588>



In [14]:
```
#checking the relation of joining bonus , candidate relocate and gender with joining status

#first we change the values of status from 'yes'/'no'  to 0/1

num = {'Joined':1, 'Not Joined':0}
hire_df.Status.replace(num, inplace=True)
hire_df["Status"]=pd.to_numeric(hire_df["Status"], downcast="integer")
```

In [15]:
```
hire_df[['Gender','Status']].groupby(['Gender']).mean()
```

Out[15]:

Gender	Status
Female	0.823985
Male	0.810796

In [16]:
```
hire_df[['Joining.Bonus','Status']].groupby(['Joining.Bonus']).mean()
```

Out[16]:

Joining.Bonus	Status
No	0.813425
Yes	0.805755

In [17]:
```
hire_df[['Candidate.relocate.actual','Status']].groupby(['Candidate.relocate.actual']).mean()
```

Out[17]:

Candidate.relocate.actual	Status
No	0.781785
Yes	1.000000

Except for Candidate.relocate.actual no other feature has significant effect on Joining status so we will only include Candidate.relocate.actual feature in model of all the binary data features

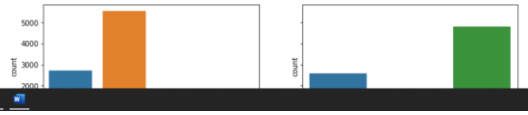
Checking the effect of other features on the target

Checking non-binary data columns

In [18]:
```
fig, axes = plt.subplots(2, 2, figsize=(12, 7), sharey=True)

sns.countplot("Offered.band", data=hire_df, ax=axes[0,0])
sns.countplot("Candidate.Source", data=hire_df, ax=axes[0,1])
sns.countplot("LOB", data=hire_df, ax=axes[1,0])
sns.countplot("Location", data=hire_df, ax=axes[1,1])
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x20284e5d518>

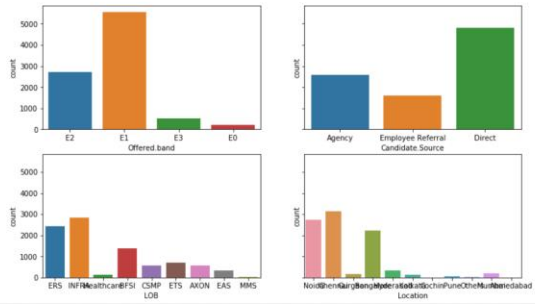


Checking non-binary data columns

In [18]:
```
fig, axes = plt.subplots(2, 2, figsize=(12, 7), sharey=True)

sns.countplot("Offered.band", data=hire_df, ax=axes[0,0])
sns.countplot("Candidate.Source", data=hire_df, ax=axes[0,1])
sns.countplot("LOB", data=hire_df, ax=axes[1,0])
sns.countplot("Location", data=hire_df, ax=axes[1,1])
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x20284e5d518>
```



In [21]:
```
hire_df[['LOB','Status']].groupby(['LOB']).mean()
```

Out[21]:

LOB	Status
AXON	0.774648
BFSI	0.758596
CSMP	0.815199
EAS	0.734104
ERS	0.781211
ETS	0.831169
Healthcare	0.822581
INFRA	0.877895
MMS	1.000000

In [22]:
```
hire_df.LOB.value_counts()
```

Out[22]:
```
INFRA         2850
ERS           2427
BFSI          1396
ETS            693
CSMP           579
AXON           568
EAS            346
Healthcare     124
MMS             15
Name: LOB, dtype: int64
```

In [23]:
```
hire_df[['Location','Status']].groupby(['Location']).mean()
```

In [19]:
```
hire_df[['Offered.band','Status']].groupby(['Offered.band']).mean()
```

Out[19]:

Offered.band	Status
E0	0.763033
E1	0.813106
E2	0.809735
E3	0.851485

In [20]:
```
hire_df[['Candidate.Source','Status']].groupby(['Candidate.Source']).mean()
```

Out[20]:

Candidate.Source	Status
Agency	0.758221
Direct	0.820112
Employee Referral	0.880124

The offered.band and candidate.Source features are important for the model because of the high rate of joining of E3 candidates and referral candidates

In [21]:
```
hire_df[['LOB','Status']].groupby(['LOB']).mean()
```

Out[21]:

LOB	Status
AXON	0.774648

In [23]:
```
hire_df[['Location','Status']].groupby(['Location']).mean()
```

Out[23]:

Location	Status
Ahmedabad	0.833333
Bangalore	0.781264
Chennai	0.789273
Cochin	0.875000
Gurgaon	0.808219
Hyderabad	0.780059
Kolkata	0.775194
Mumbai	0.893401
Noida	0.866202
Others	1.000000
Pune	0.791667

I won't use location for training the model

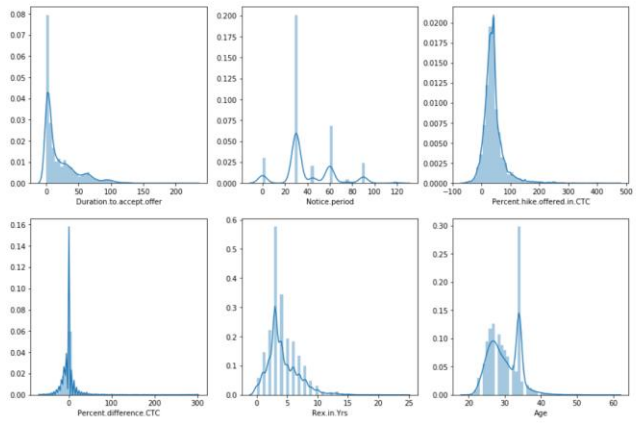
Checking continous variables

The important continous variables to check are :- Duration.to.accept.offer , Notice.period , Percent.hike.offered.in.CTC , Percent.difference.CTC , Rex.in.Yrs and Age .

In [24]:
```
fig, axes = plt.subplots(2,3, figsize=(15, 10))
```

```
g is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[n
p.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2028540abe0>
```



In [25]:
```
hire_df[['Percent.hike.offered.in.CTC','Percent.difference.CTC','Status']].groupby('Status').mean()
```

Out[25]:

Status	Percent.hike.offered.in.CTC	Percent.difference.CTC
0	38.588460	-2.929298
1	41.147158	-1.263402

The Percent.difference.CTC is important to consider for the model and we can leave the Percent.hike.offered.in.CTC

In [26]:
```
hire_df[['Duration.to.accept.offer','Notice.period',"Rex.in.Yrs","Age",'Status']].groupby('Status').mean()
```

Out[26]:

Status	Duration.to.accept.offer	Notice.period	Rex.in.Yrs	Age
0	24.956599	48.192628	4.439358	29.517836
1	20.617687	37.233461	4.193002	30.004647

and also the Duration.to.accept.offer and Notice.period columns seems to be effecting the status column data so we will include these two featues too

Dropping irrelevent features of the dataframe

After analizing all the different types of variables i have decided to drop the following columns :- 'SLNO', 'Candidate.Ref', 'DOJ.Extended','Joining.Bonus','Gender','Rex.in.Yrs', 'Location', 'Age'

In [27]:
```
hire_df.drop(['SLNO', 'Candidate.Ref', 'DOJ.Extended', 'Joining.Bonus','Gender', 'Rex.in.Yrs', 'Location', 'Age'], axis=1, inpla
```

Data Preprocessing

In [29]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.preprocessing import MinMaxScaler
```

In [30]:
```
cat_feat=['Candidate.relocate.actual','Offered.band','Candidate.Source','LOB']
X = pd.get_dummies(hire_df, columns=cat_feat, drop_first=True)
```

In [31]:
```
sc = MinMaxScaler()
a = sc.fit_transform(hire_df[['Duration.to.accept.offer']])
b = sc.fit_transform(hire_df[['Notice.period']])
c = sc.fit_transform(hire_df[['Pecent.hike.expected.in.CTC']])
d = sc.fit_transform(hire_df[['Percent.hike.offered.in.CTC']])
e = sc.fit_transform(hire_df[['Percent.difference.CTC']])
```

In [32]:
```
X["Duration.to.accept.offer"]=a
X["Notice.period"]=b
X["Pecent.hike.expected.in.CTC"]=c
X["Percent.hike.offered.in.CTC"]=d
X["Percent.difference.CTC"]=e
```

In [33]:
```
X.shape
```

Out[33]:
```
(8998, 20)
```

In [34]:
```
X.head()
```

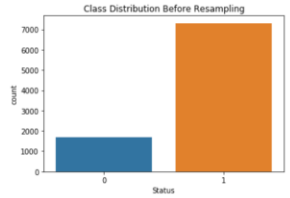
Out[34]:

	Duration.to.accept.offer	Notice.period	Pecent.hike.expected.in.CTC	Percent.hike.offered.in.CTC	Percent.difference.CTC	Status	Candidate.relocate.actual_Yes	C
0	0.062500	0.250	0.112086	0.138525	0.299861	1	0	

Sampling

In [35]:
```
sns.countplot('Status', data=hire_df).set_title('Class Distribution Before Resampling')
```

Out[35]:
```
Text(0.5,1,'Class Distribution Before Resampling')
```



In [36]:
```
X_no = X[X.Status == 0]
X_yes = X[X.Status == 1]
```

In [37]:
```
print(len(X_no),len(X_yes))
```

```
1682 7316
```

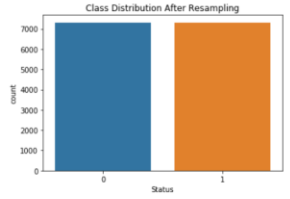
In [38]:
```
X_no_upsampled = X_no.sample(n=len(X_yes), replace=True, random_state=42)
print(len(X_no_upsampled))
```

```
7316
```

In [39]:
```
X_upsampled = X_yes.append(X_no_upsampled).reset_index(drop=True)
```

In [40]:
```
sns.countplot('Status', data=X_upsampled).set_title('Class Distribution After Resampling')
```

Out[40]:
```
Text(0.5,1,'Class Distribution After Resampling')
```

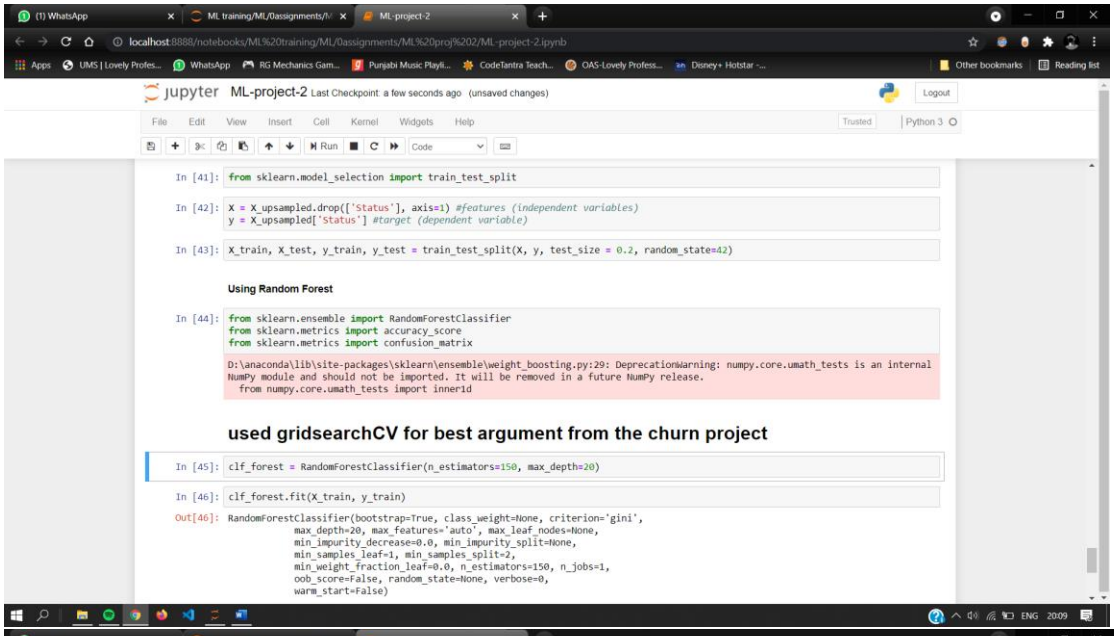


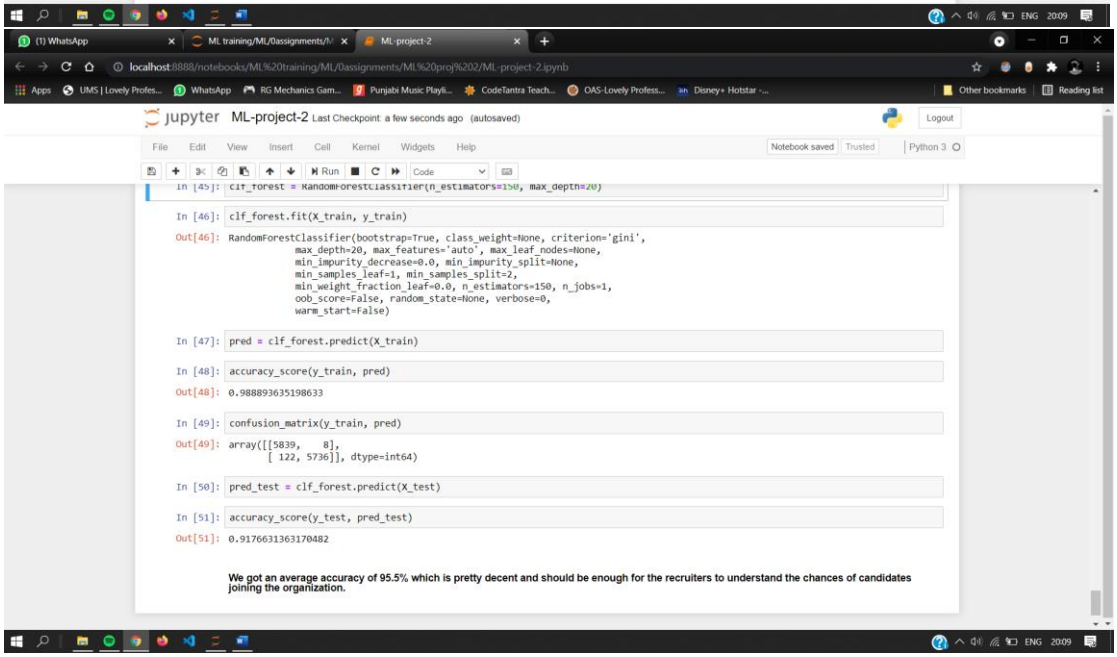
ML Model

In [41]:
```
from sklearn.model_selection import train_test_split
```

In [42]:
```
X = X_upsampled.drop(['Status'], axis=1) #features (independent variables)
y = X_upsampled['Status'] #target (dependent variable)
```

In [43]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=42)
```


ML-project-2 Last Checkpoint: a few seconds ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [41]: from sklearn.model_selection import train_test_split
In [42]: X = X_upsampled.drop(['Status'], axis=1) #features (independent variables)
y = X_upsampled['Status'] #target (dependent variable)
In [43]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=42)
Using Random Forest
In [44]: from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
D:\anaconda\lib\site-packages\sklearn\ensemble\weight_boosting.py:29: DeprecationWarning: numpy.core.umath_tests is an internal NumPy module and should not be imported. It will be removed in a future NumPy release.
from numpy.core.umath_tests import inner1d
used gridsearchCV for best argument from the churn project
In [45]: clf_forest = RandomForestClassifier(n_estimators=150, max_depth=20)
In [46]: clf_forest.fit(X_train, y_train)
Out[46]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
max_depth=20, 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_fraction_leaf=0.0, n_estimators=150, n_jobs=1,
oob_score=False, random_state=None, verbose=0,
warm_start=False)


ML-project-2 Last Checkpoint: a few seconds ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Notebook saved
Trusted
Python 3
Run
Code
In [46]: clf_forest.fit(X_train, y_train)
Out[46]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
max_depth=20, 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_fraction_leaf=0.0, n_estimators=150, n_jobs=1,
oob_score=False, random_state=None, verbose=0,
warm_start=False)
In [47]: pred = clf_forest.predict(X_train)
In [48]: accuracy_score(y_train, pred)
Out[48]: 0.988893635198633
In [49]: confusion_matrix(y_train, pred)
Out[49]: array([[5839, 8],
[122, 5736]], dtype=int64)
In [50]: pred_test = clf_forest.predict(X_test)
In [51]: accuracy_score(y_test, pred_test)
Out[51]: 0.9176631363170482
We got an average accuracy of 95.5% which is pretty decent and should be enough for the recruiters to understand the chances of candidates joining the organization.