

**NAME**: MUHAMMAD ADNAN AYUB

**ID**  : F2018065063

**COURSE:** DATA SCIENCE TECHNOLOGIES

**SUBJECT**: PROJECT DOCUMENTATION

**TO**  : MAZHAR JAVED AWAN

**PROJECT DOCUMENTATION**

**This** documentation is on Data Science Project that I built. I have selected a Pakistan Job Market data set from 2019 to 2021 and did some data analysis, applied some pandas and numpy operations, also plotted many charts by using matpolit libraries and seaborns to compare different labels features of data set. After doing all these things I finally Build a Linear Regression, Random Forest and Logistic Regression Model by creating dummy variables for logistic Regression because Pak job market data set have many categorical variables so had to convert them into numeric ones.

First I inserted libraries that I have to use to build the model and do some data analysis stuff.

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**from sklearn.metrics import accuracy\_score**

**from sklearn.metrics import classification\_report**

**from sklearn.metrics import confusion\_matrix**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn import linear\_model**

**%matplotlib inline**

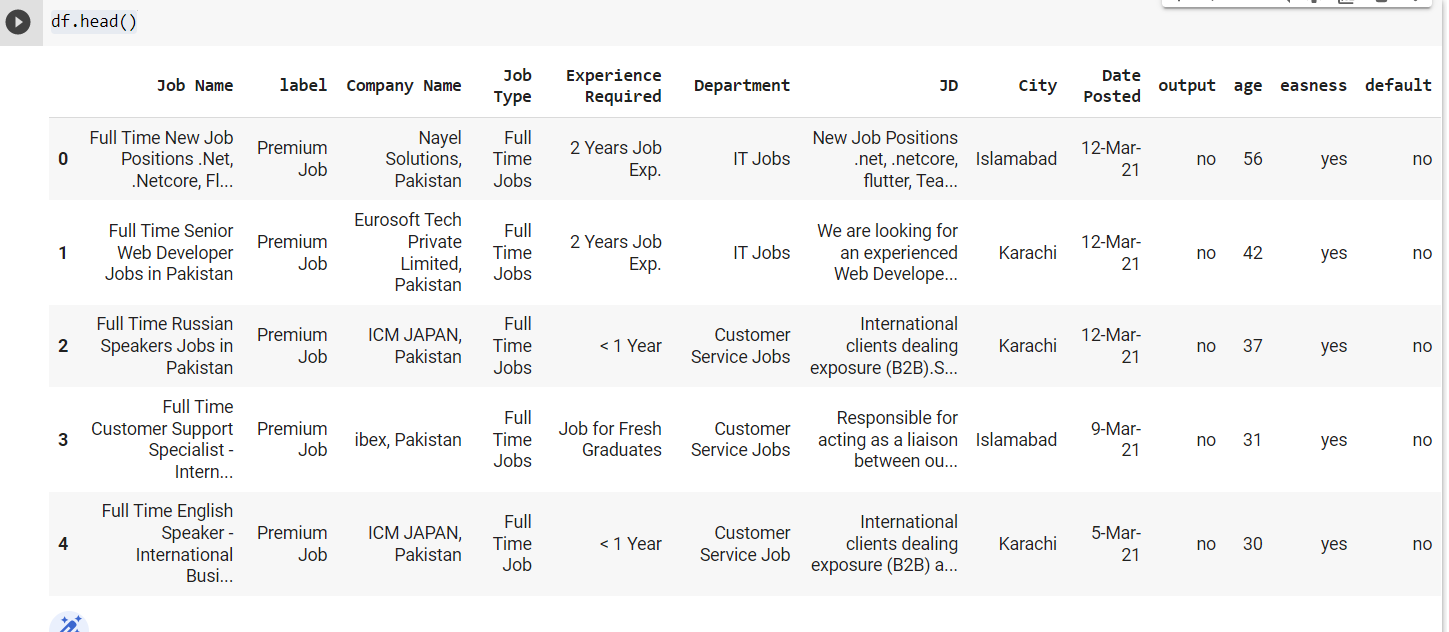
**sns.set\_style('darkgrid')**

Then I loaded my Pakistan Job Market Dataset:

**df = pd.read\_csv('/content/Pakistan Available Job Dec 19 - Mar-21.csv')**

Then I did Data Analysis by checking the first 5 rows of Dataset:

**df.head()**

****

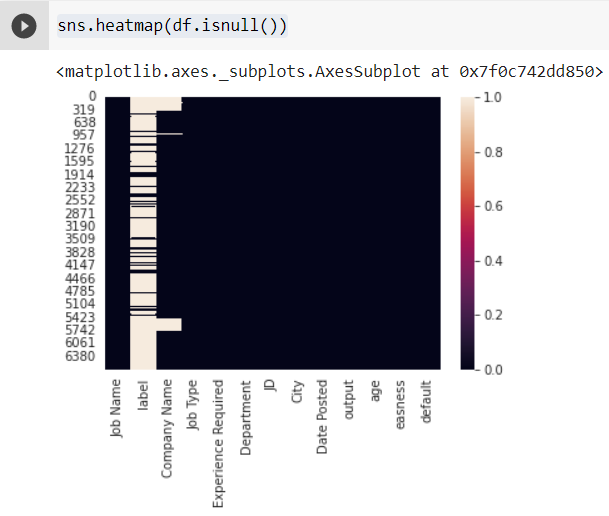
Also Checked the last 5 as well:

**df.tail()**



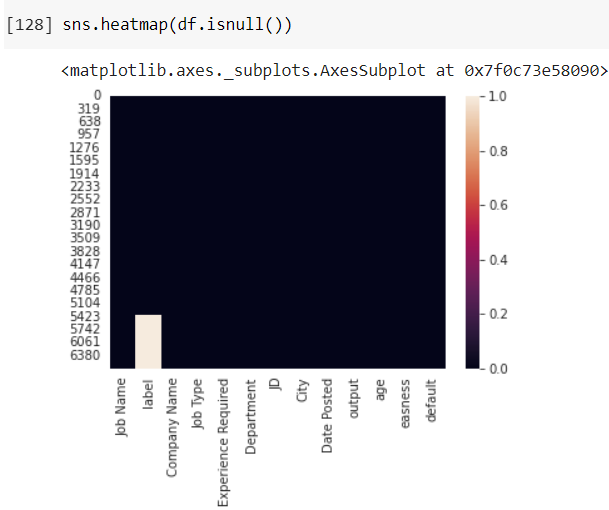
Then we plot a Heatmap to check if our data set has missing values:

**sns.heatmap(df.isnull())**

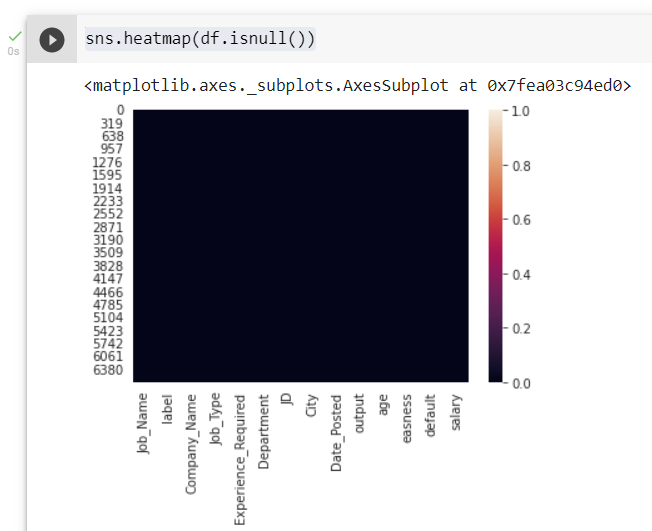


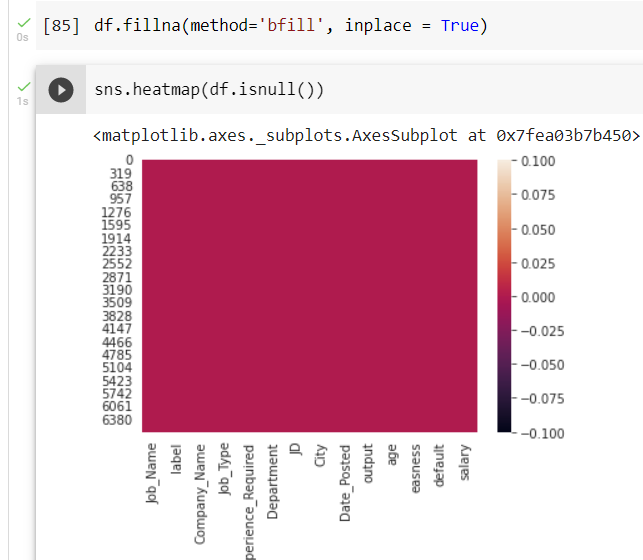
Then We fill the missing values:

**df.fillna(method='bfill', inplace = True)**

****

**sns.heatmap(df.isnull())**

****



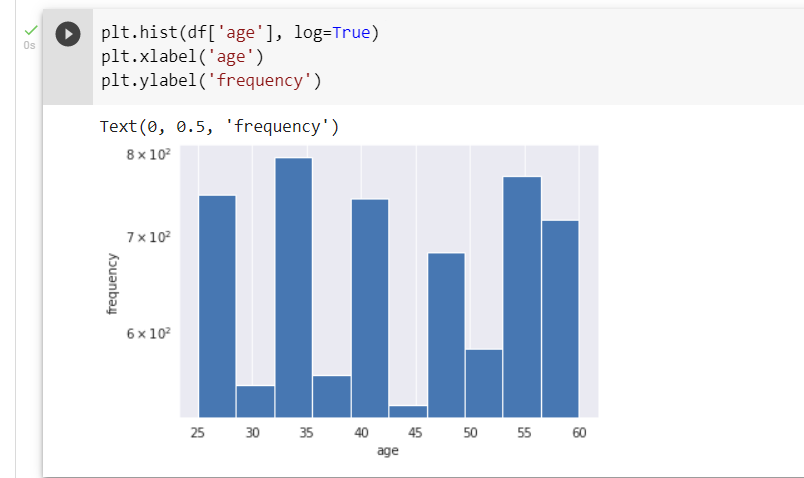
**df.columns**



**plt.hist(df['age'], log=True)**

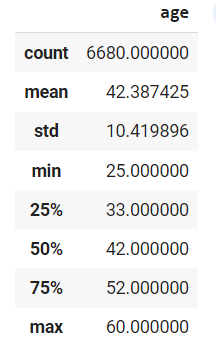
**plt.xlabel('age')**

**plt.ylabel('frequency')**

****

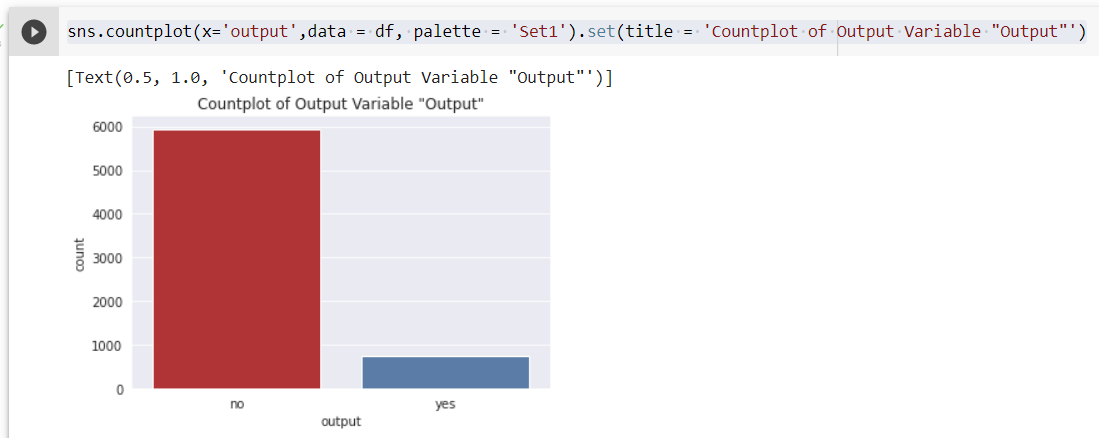
Then We perform statistics on age requirement feature:

**df.describe()**

****

**Finding How many Jobs have Highest Salary and low salary from our output variable:**

**sns.countplot(x='output',data = df, palette = 'Set1').set(title = 'Countplot of Output Variable "Output"')**

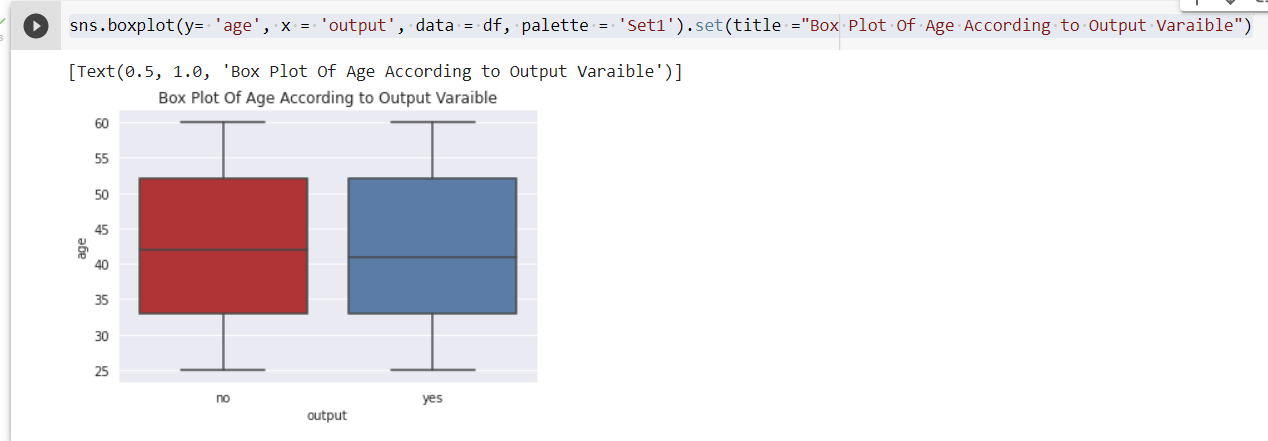
****

**Measuring Counterplot Of Output variable with easness:**

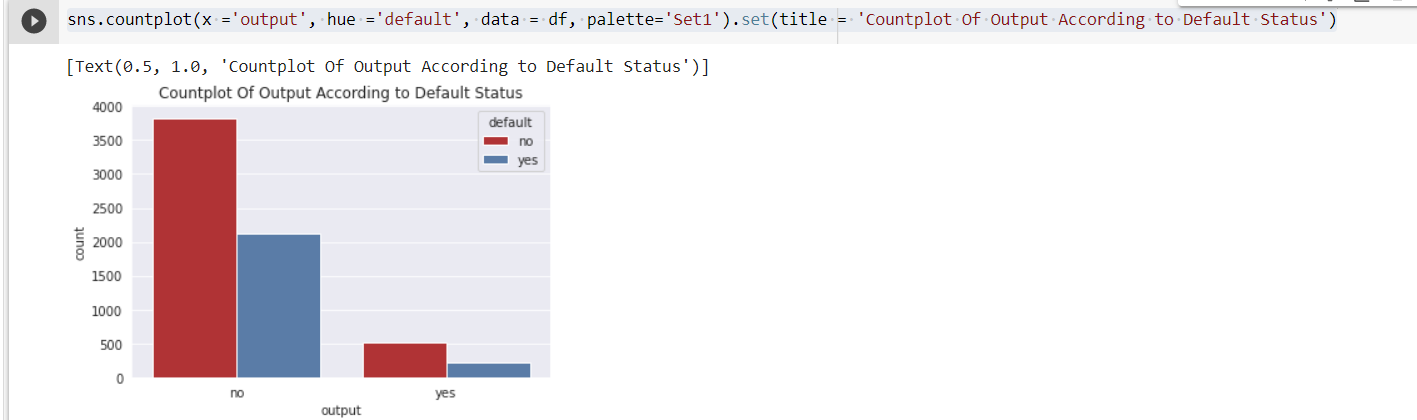
**sns.countplot(x ='output', hue ='easness', data =df, palette='Set1').set(title = "Countplot Of Output And Easness ")**

****

**sns.boxplot(y= 'age', x = 'output', data = df, palette = 'Set1').set(title ="Box Plot Of Age According to Output Varaible")**



**sns.countplot(x ='output', hue ='default', data = df, palette='Set1').set(title = 'Countplot Of Output According to Default Status')**

****

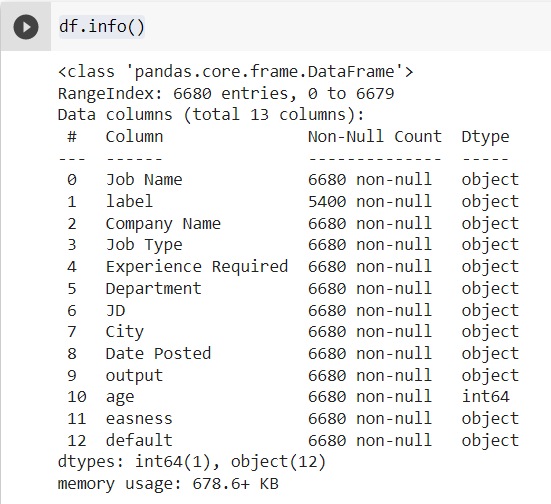
**We can see from countplots of the output low and high Salary (y) and easniess status of jobs that whether or not a job has ease may have an impact on the likelihood of a yes. From the next plot, we can make an educated guess that age will not have a significant impact on Jobs.**

**df.shape**

****

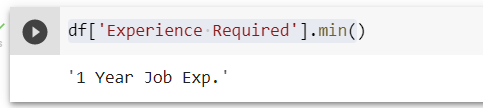
**Checking the Data Types:**

**df.info()**

****

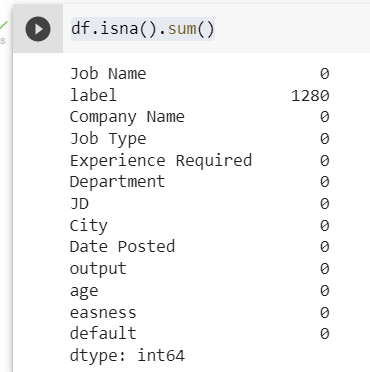
**Finding the Minimum Experience Required For a Job:**

**df['Experience Required'].min()**

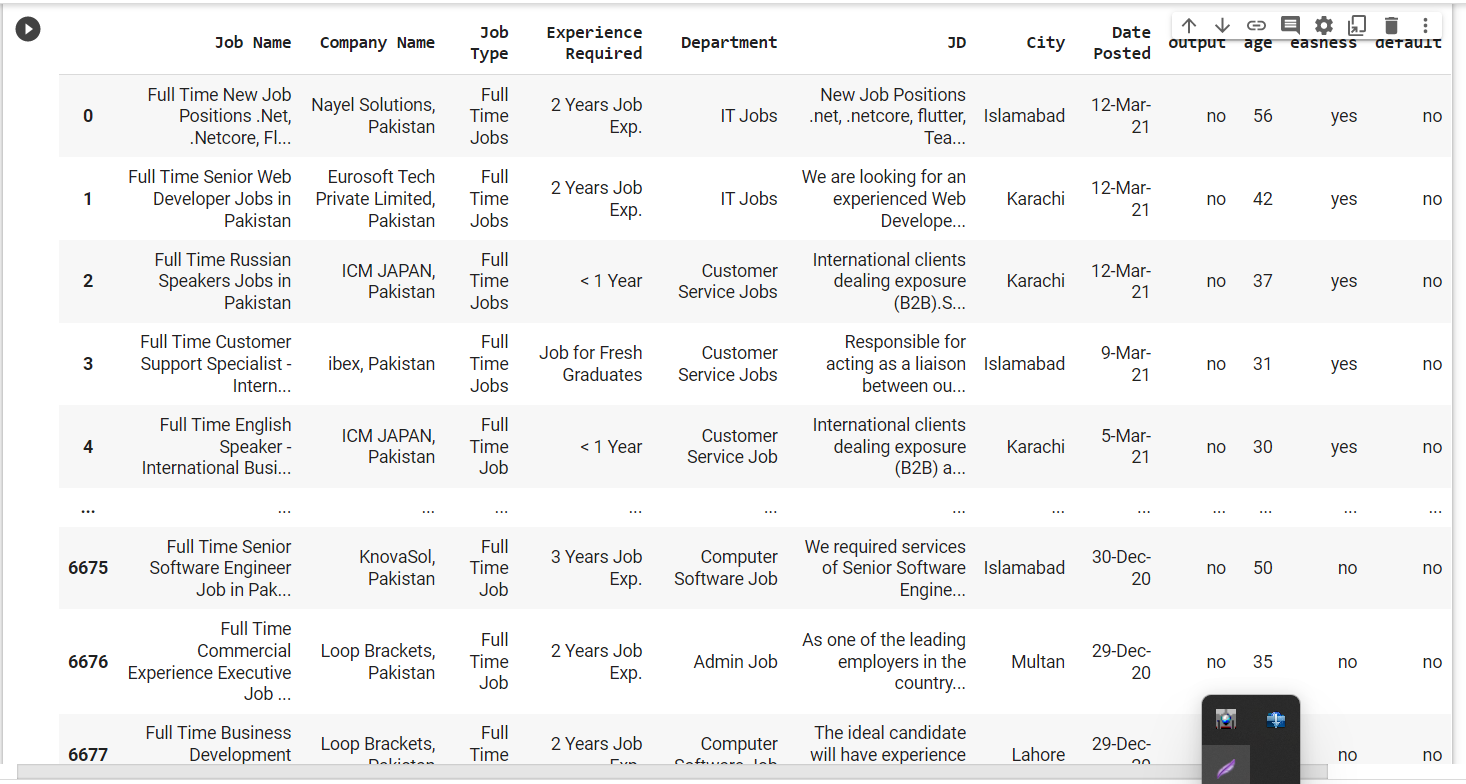
****

Check Null **values:**

**df.isna().sum()**

****

**df.dropna(axis=1)**

****

**How Many jobs available in Pakistan:**

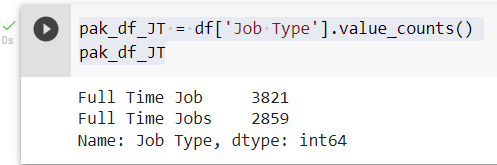
**df['Job Name'].count()**

**(6680)**

**Total Job Types Counting:**

**pak\_df\_JT = df['Job Type'].value\_counts()**

**pak\_df\_JT**

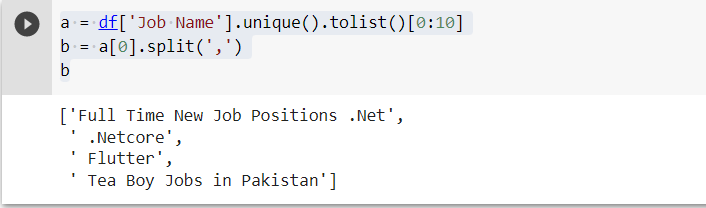
****

**Most Demanding Jobs in Pakistan:**

**a = df['Job Name'].unique().tolist()[0:10]**

**b = a[0].split(',')**

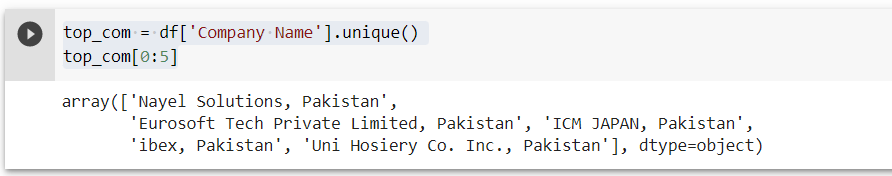
**b**



**Top 5 Companies who provide jobs:**

**top\_com = df['Company Name'].unique()**

**top\_com[0:5]**

****

**Data Scientist Jobs in Pakistan:**

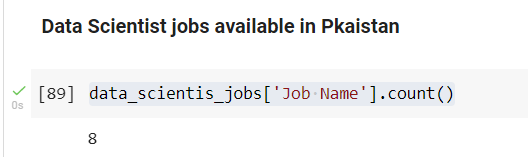
**data\_scientis\_jobs = df[df['Job Name'].str.contains('Data Scientist')]**

**data\_scientis\_jobs**

****

**Data Scientist jobs available in Pkaistan:**

**data\_scientis\_jobs['Job Name'].count()**

****

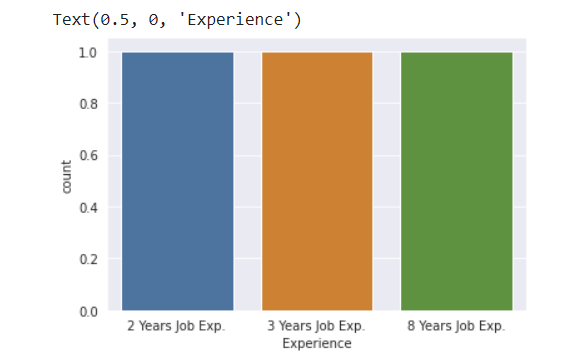
**MATPOLIT AND SEABORN PLOTTING:**

**Minimum Experience for Applying a data scientist jobs:**

**e = data\_scientis\_jobs['Experience Required'].unique()**

**sns.countplot(x = e, data = df)**

**plt.xlabel('Experience')**

****

**c = data\_scientis\_jobs['City'].unique()**

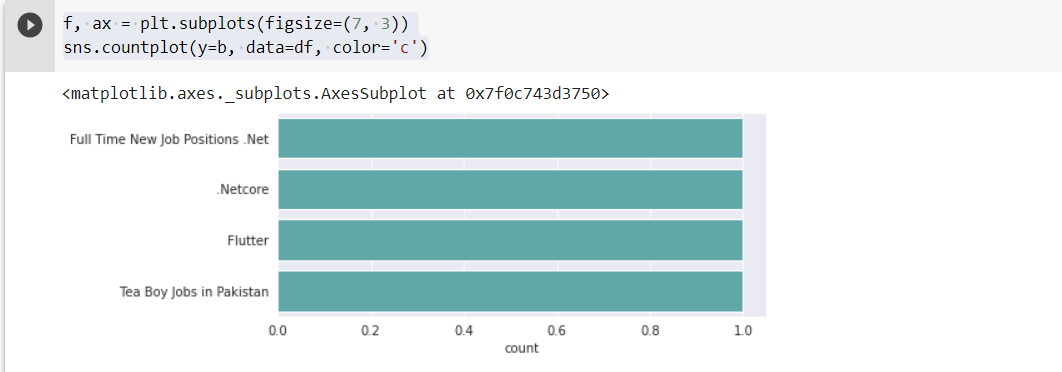
**sns.countplot(x = c, data = df)**

**plt.xlabel('City')**

****

**f, ax = plt.subplots(figsize=(7, 3))**

**sns.countplot(y=b, data=df, color='c')**

****

**plt.figure(figsize=(10,10))**

**res=sns.barplot(x=pak\_df\_JT, y=pak\_df\_JT.index)**

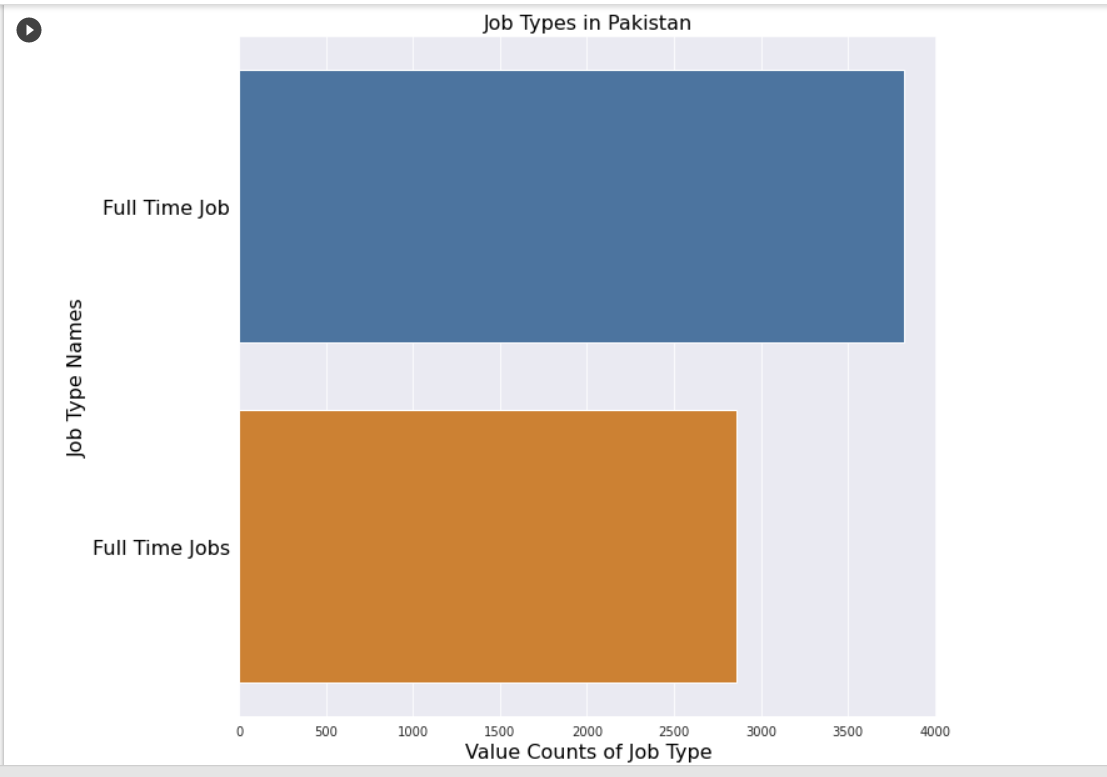
**res.set\_yticklabels(res.get\_ymajorticklabels(), fontsize = 16, color='black')**

**plt.xlabel('Value Counts of Job Type',fontsize = 16, color='black')**

**plt.ylabel('Job Type Names',fontsize = 16, color='black')**

**plt.title('Job Types in Pakistan',fontsize = 16, color='black')**

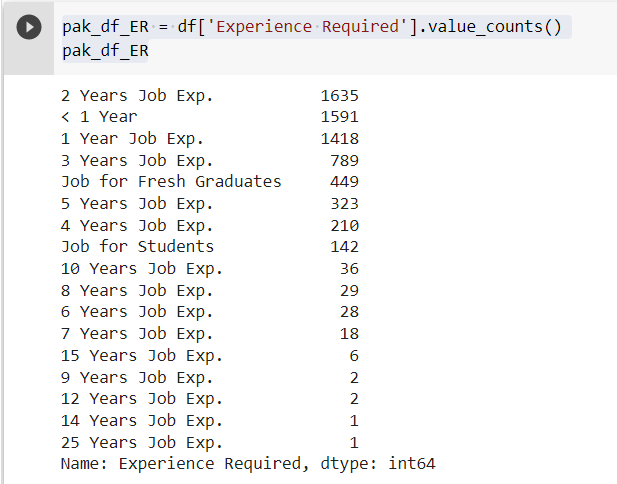
**plt.show()**

****

**How many years of experience is required for job:**

**pak\_df\_ER = df['Experience Required'].value\_counts()**

**pak\_df\_ER**

****

**plt.figure(figsize=(10,10))**

**res=sns.barplot(x=pak\_df\_ER, y=pak\_df\_ER.index)**

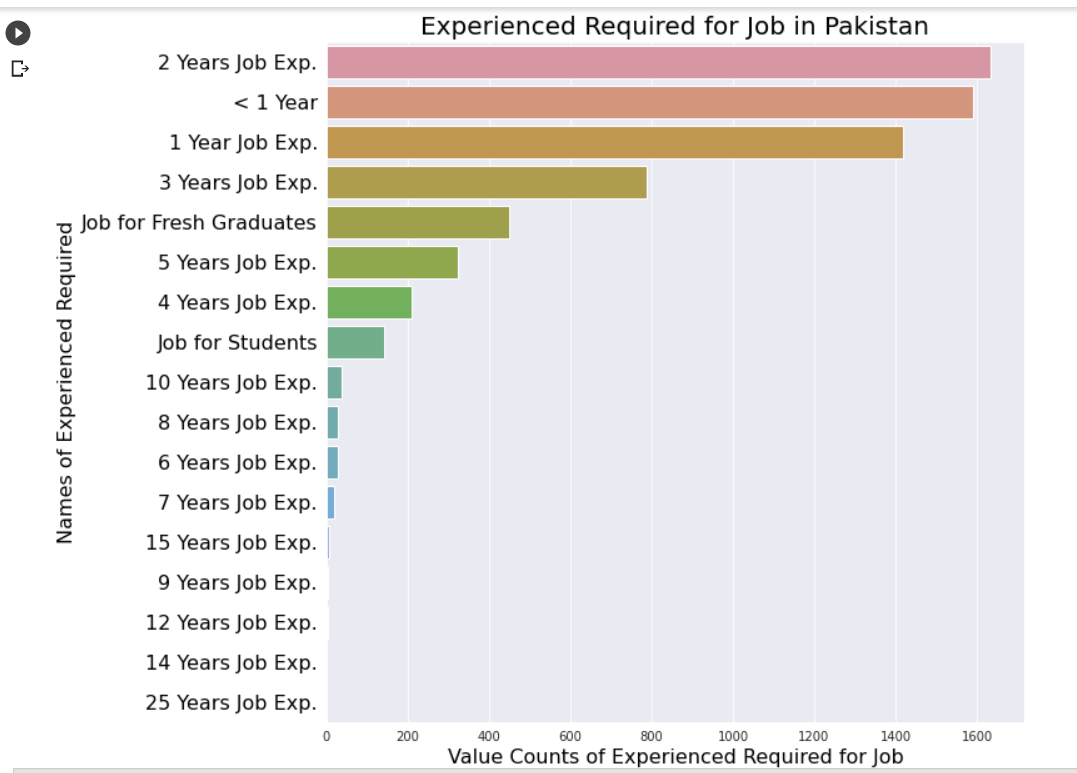
**res.set\_yticklabels(res.get\_ymajorticklabels(), fontsize = 16, color='black')**

**plt.xlabel('Value Counts of Experienced Required for Job', fontsize = 16, color='black')**

**plt.ylabel('Names of Experienced Required', fontsize = 16, color='black')**

**plt.title('Experienced Required for Job in Pakistan', fontsize = 20, color='black')**

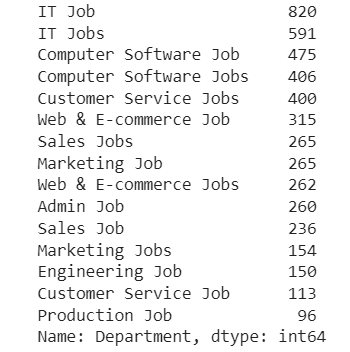
**plt.show()**

****

**How Many Jobs Are Available In Each Departments:**

**pak\_df\_Dept = df['Department'].value\_counts().head(15)**

**pak\_df\_Dept**

****

**plt.figure(figsize=(10,8))**

**res=sns.barplot(x=pak\_df\_Dept, y=pak\_df\_Dept.index)**

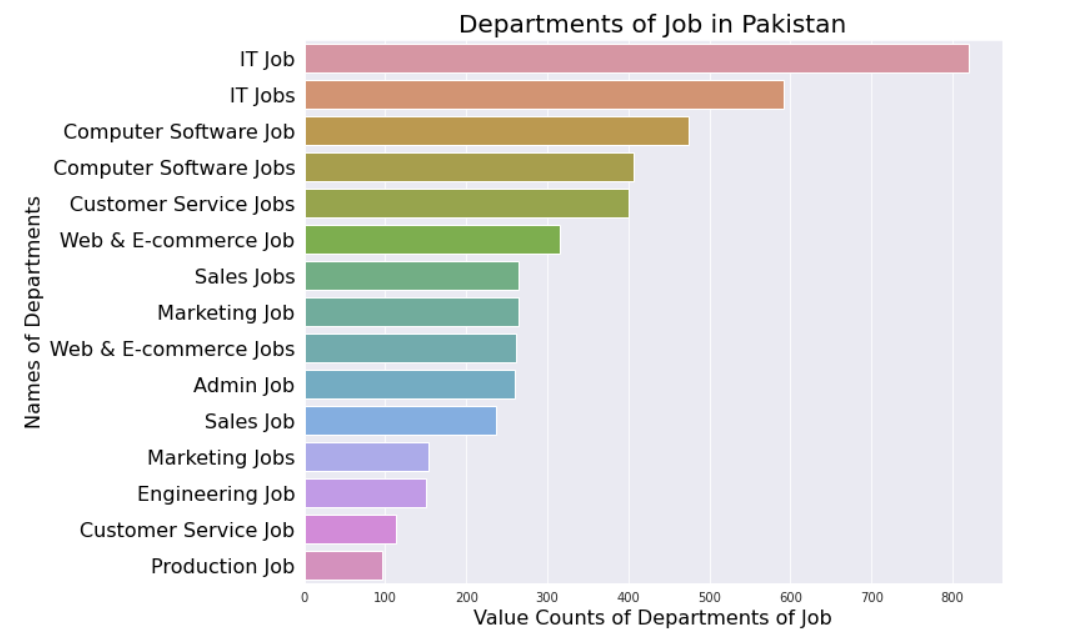
**res.set\_yticklabels(res.get\_ymajorticklabels(), fontsize = 16, color='black')**

**plt.xlabel('Value Counts of Departments of Job', fontsize = 16, color='black')**

**plt.ylabel('Names of Departments', fontsize = 16, color='black')**

**plt.title('Departments of Job in Pakistan', fontsize = 20, color='black')**

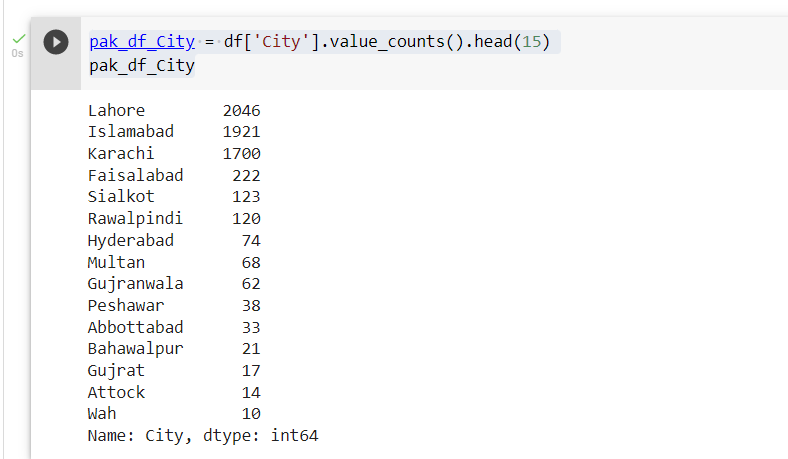
**plt.show()**



# How many Jobs are available in each specific city:

**pak\_df\_City = df['City'].value\_counts().head(15)**

**pak\_df\_City**

****

**plt.figure(figsize=(10,8))**

**res=sns.barplot(x=pak\_df\_City, y=pak\_df\_City.index)**

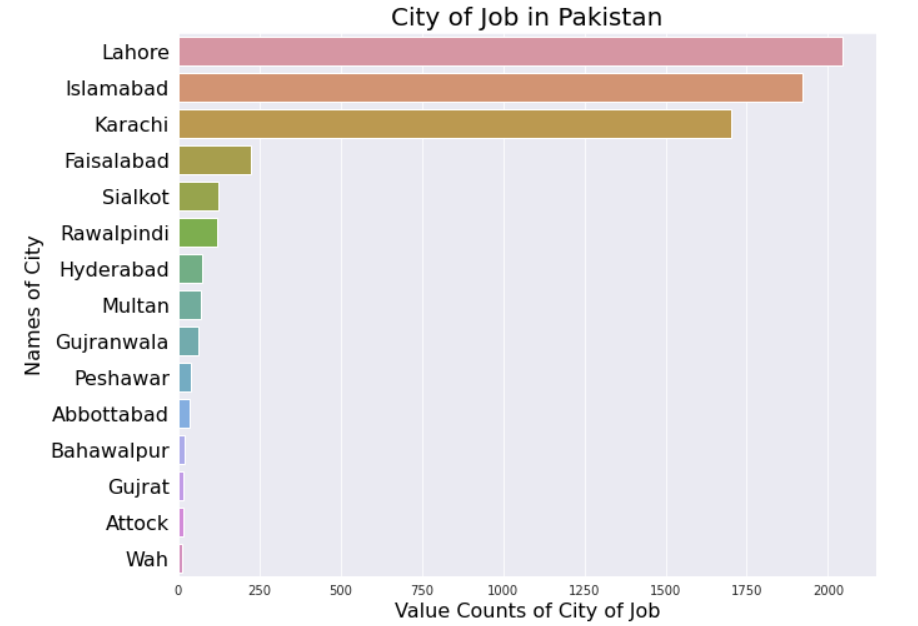
**res.set\_yticklabels(res.get\_ymajorticklabels(), fontsize = 16, color='black')**

**plt.xlabel('Value Counts of City of Job', fontsize = 16, color='black')**

**plt.ylabel('Names of City', fontsize = 16, color='black')**

**plt.title('City of Job in Pakistan', fontsize = 20, color='black')**

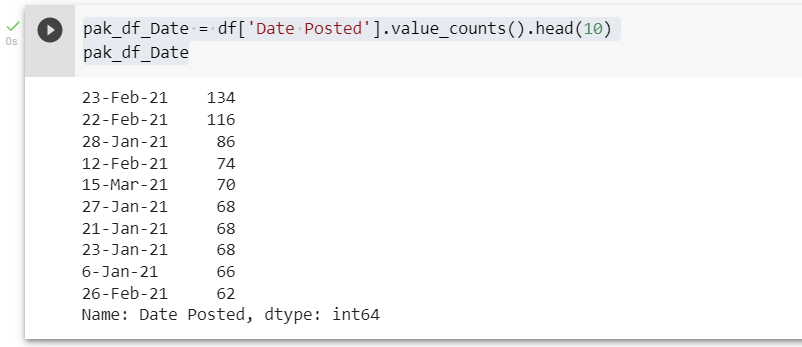
**plt.show()**

****

# Finding how many Jobs are Posted in each dates:

**pak\_df\_Date = df['Date Posted'].value\_counts().head(10)**

**pak\_df\_Date**

****

plt.figure(figsize=(10,8))

res=sns.barplot(x=pak\_df\_Date, y=pak\_df\_Date.index)

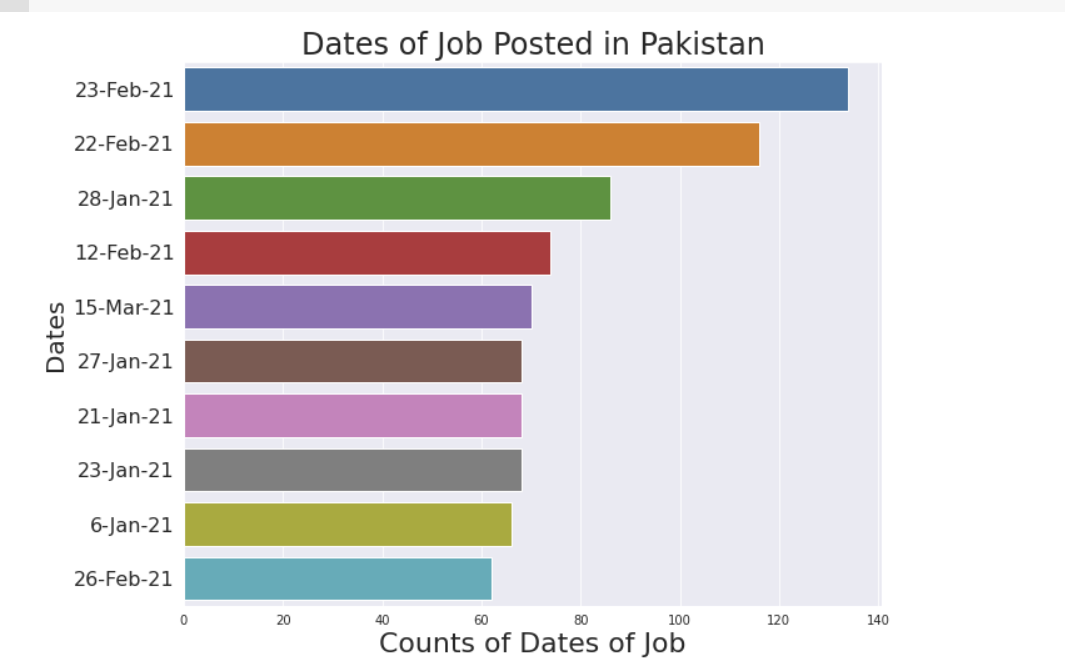
res.set\_yticklabels(res.get\_ymajorticklabels(), fontsize = 16)

plt.xlabel('Counts of Dates of Job', fontsize = 22)

plt.ylabel('Dates', fontsize = 20)

plt.title('Dates of Job Posted in Pakistan', fontsize = 24)

plt.show()

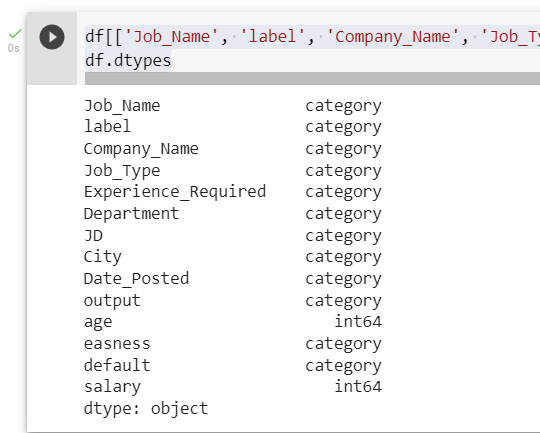


**LINEAR REGRESSION MODEL:**

**FIRST CONVERTING NUM VARS TO A CATEGORIACAL VALUES:**

**df[['Job\_Name', 'label', 'Company\_Name', 'Job\_Type', 'Experience\_Required', 'Department', 'JD', 'City', 'Date\_Posted', 'output', 'easness', 'default']] = df[['Job\_Name', 'label', 'Company\_Name', 'Job\_Type', 'Experience\_Required', 'Department', 'JD', 'City', 'Date\_Posted', 'output', 'easness', 'default']].astype('category')**

**df.dtypes**

****

**Now Converting category labels into numerical using LabelEncoder():**

**from sklearn.preprocessing import LabelEncoder**

**label = LabelEncoder()**

**label.fit(df.Job\_Name.drop\_duplicates())**

**df.Job\_Name = label.transform(df.Job\_Name)**

**label.fit(df.label.drop\_duplicates())**

**df.label = label.transform(df.label)**

**label.fit(df.Company\_Name.drop\_duplicates())**

**df.Company\_Name = label.transform(df.Company\_Name)**

**label.fit(df.Job\_Type.drop\_duplicates())**

**df.Job\_Type = label.transform(df.Job\_Type)**

**label.fit(df.Experience\_Required.drop\_duplicates())**

**df.Experience\_Required = label.transform(df.Experience\_Required)**

**label.fit(df.Department.drop\_duplicates())**

**df.Department = label.transform(df.Department)**

**label.fit(df.JD.drop\_duplicates())**

**df.JD = label.transform(df.JD)**

**label.fit(df.City.drop\_duplicates())**

**df.City = label.transform(df.City)**

**label.fit(df.Date\_Posted.drop\_duplicates())**

**df.Date\_Posted = label.transform(df.Date\_Posted)**

**label.fit(df.output.drop\_duplicates())**

**df.output = label.transform(df.output)**

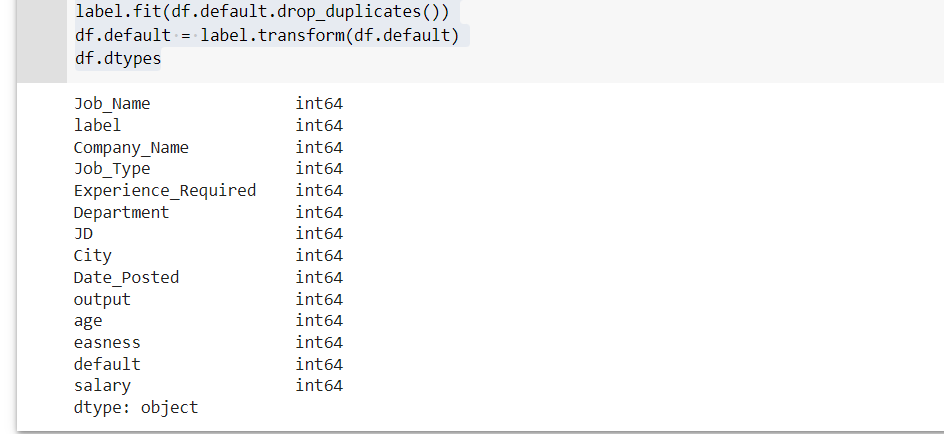
**label.fit(df.easness.drop\_duplicates())**

**df.easness = label.transform(df.easness)**

**label.fit(df.default.drop\_duplicates())**

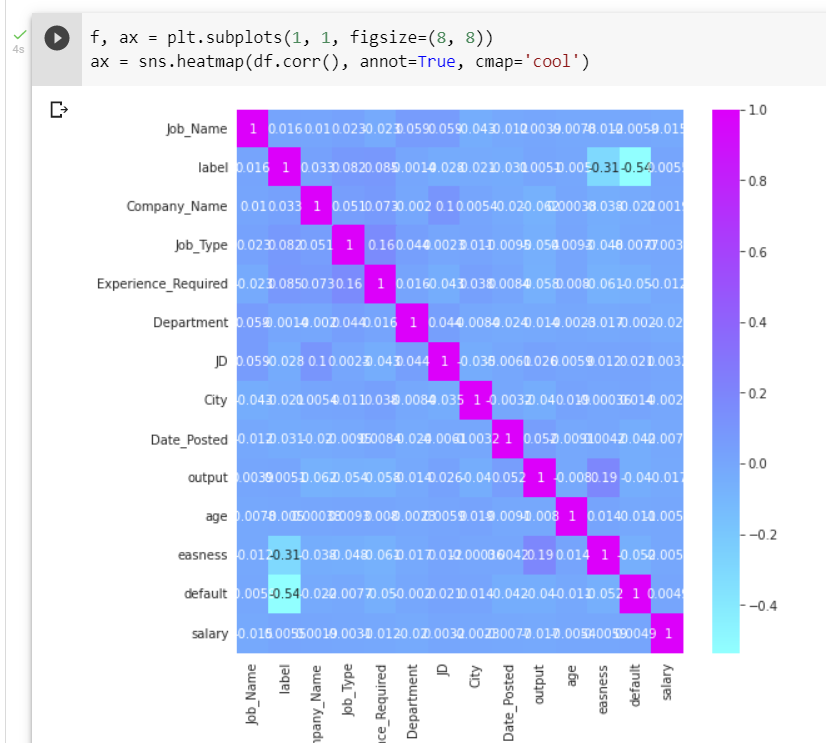
**df.default = label.transform(df.default)**

**df.dtypes**

****

**f, ax = plt.subplots(1, 1, figsize=(11, 11))**

**ax = sns.heatmap(df.corr(), annot=True, cmap='cool')**

****

**from sklearn.model\_selection import train\_test\_split as holdout**

**from sklearn.linear\_model import LinearRegression**

**from sklearn import metrics**

**x = df.drop(['age'], axis = 1)**

**y = df['salary']**

**x\_train, x\_test, y\_train, y\_test = holdout(x, y, test\_size=0.2, random\_state=0)**

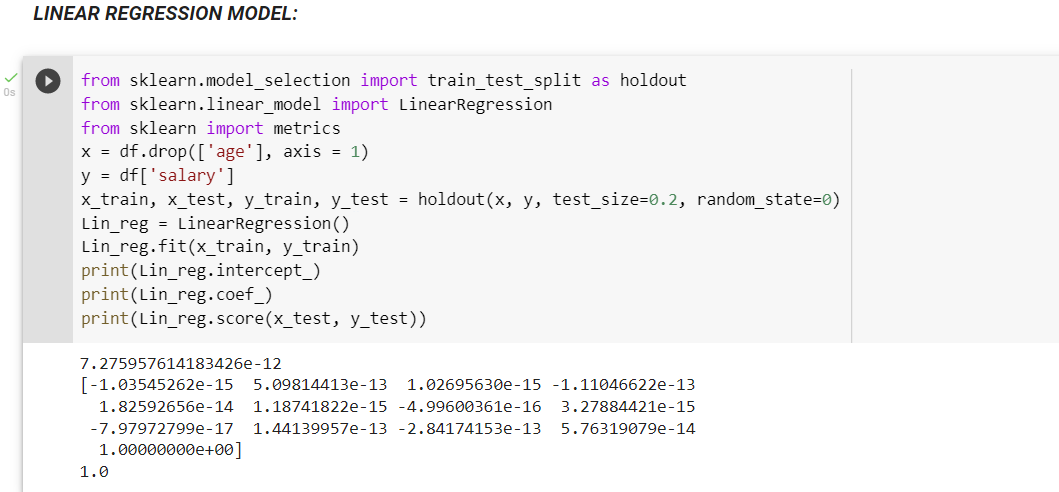
**Lin\_reg = LinearRegression()**

**Lin\_reg.fit(x\_train, y\_train)**

**print(Lin\_reg.intercept\_)**

**print(Lin\_reg.coef\_)**

**print(Lin\_reg.score(x\_test, y\_test))**

****

**RANDOM FOREST REGRESSION ALGORITHM:**

**from sklearn.ensemble import RandomForestRegressor as rfr**

**x = df.drop(['age'], axis=1)**

**y = df.salary**

**Rfr = rfr(n\_estimators = 100, criterion = 'mse', random\_state = 1, n\_jobs = -1)**

**Rfr.fit(x\_train,y\_train)**

**x\_train\_pred = Rfr.predict(x\_train)**

**x\_test\_pred = Rfr.predict(x\_test)**

**# Printing the train data based on scores:**

**print('Mean square e train data: %.3f, MSE test data: %.3f' %**

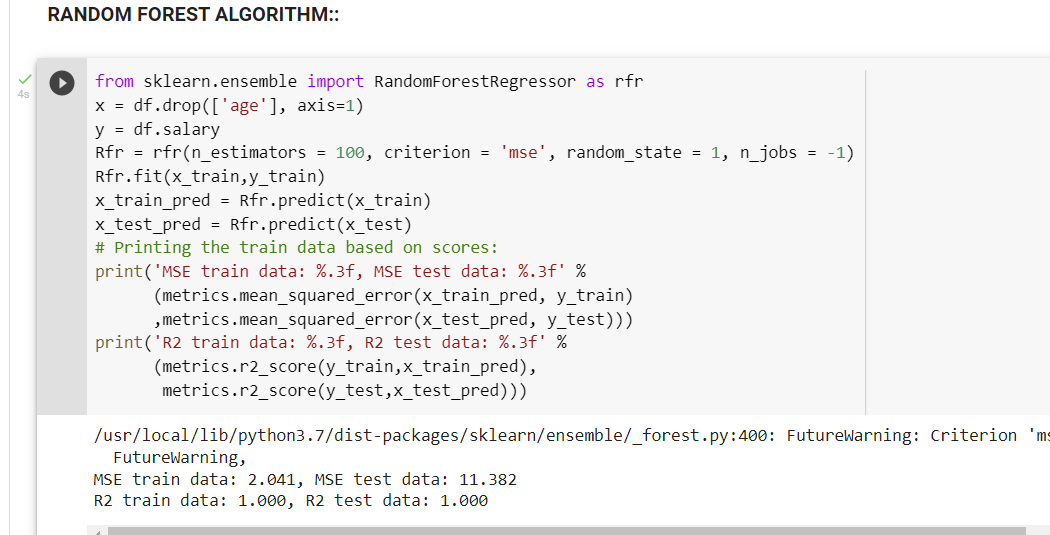
**(metrics.mean\_squared\_error(x\_train\_pred, y\_train)**

**,metrics.mean\_squared\_error(x\_test\_pred, y\_test)))**

**print('Training Data for: %.3f, R2 test data: %.3' %**

**(metrics.r2\_score(y\_train,x\_train\_pred),**

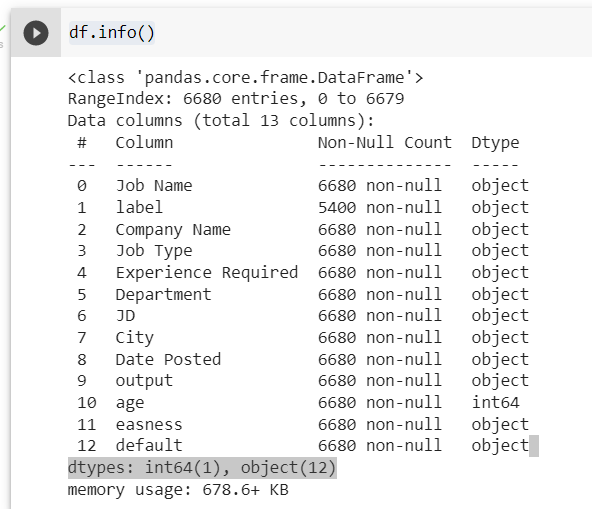
**metrics.r2\_score(y\_test,x\_test\_pred)))**

****

**Now I am Going To Build A Logistic Regression Model:**

**First we will check for categorical and numeric variables in our dataset:**

**df.info()**

****

**Now we see that we have 12 objects and need to be converted into numeric ones so that we can predict on the basis of those variables either True or False and 0 or 1.**

**For that we are converting categorical variables to a Dummy variables:**

**#dummying all Categorical variables I'll be using**

**# Job Name**

**Job\_Name = pd.get\_dummies(df['Job Name'],drop\_first=True)**

**#label**

**label = pd.get\_dummies(df['label'],drop\_first=True)**

**#Company**

**Company\_Name = pd.get\_dummies(df['Company Name'],drop\_first=True)**

**#Job Type**

**Job\_Type = pd.get\_dummies(df['Job Type'],drop\_first=True)**

**#Experience**

**Experience\_Required = pd.get\_dummies(df['Experience Required'],drop\_first=True)**

**#Department**

**Department = pd.get\_dummies(df['Department'],drop\_first=True)**

**#JD**

**JD = pd.get\_dummies(df['JD'],drop\_first=True)**

**#City**

**City = pd.get\_dummies(df['City'],drop\_first=True)**

**#Date Posted**

**Date\_Posted = pd.get\_dummies(df['Date Posted'],drop\_first=True)**

**#easness**

**easness = pd.get\_dummies(df['easness'],drop\_first=True)**

**# dropping Columns not needed**

**# dropping all columns being replaced with dummies**

**# default column based of conclusions in data**

**df.drop(['default', 'Job Name', 'label', 'Company Name','Job Type','Experience Required','Department','JD','City','Date Posted','easness'], axis = 1, inplace = True)**

**#Now Concatenating new columns with dummy variables to df**

**df = pd.concat([df, Job\_Name,label,Company\_Name,Job\_Type,Experience\_Required,JD,City,Date\_Posted,easness], axis = 1)**

**NOW WE CAN SEE THE NEW COLUMNS:**

**print(df.columns)**

****

**Now converting Yes or No to 1 and 0 Respectively:**

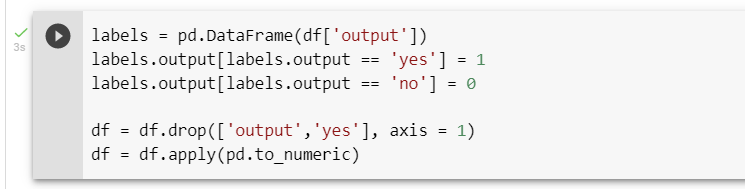
**labels = pd.DataFrame(df['output'])**

**labels.output[labels.output == 'yes'] = 1**

**labels.output[labels.output == 'no'] = 0**

**df = df.drop(['output','yes'], axis = 1)**

**df = df.apply(pd.to\_numeric)**

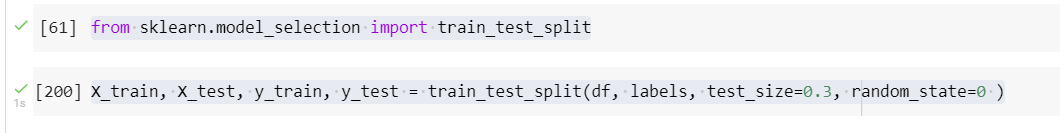
****

**BUILDING A LOGISTIC REGRESSION MODEL:**

Data Preparation: Train, Test, Split We will now split our data into training data sets and test data sets:

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(df, labels, test\_size=0.3, random\_state=0 )**

****

**Building the Model Now that we have our training and test data sets, we can train our model and make predictions:**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn import linear\_model**

**# LOGISTIC REGRESSION**

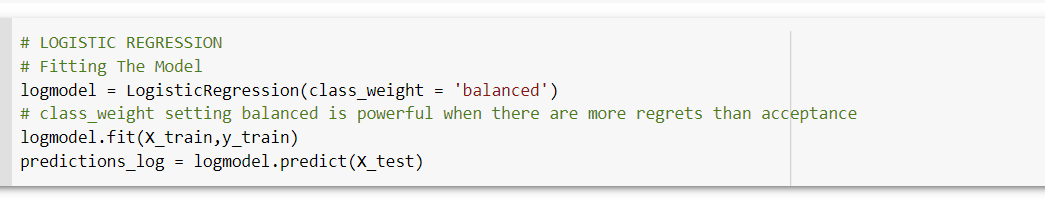
**# Fitting The Model**

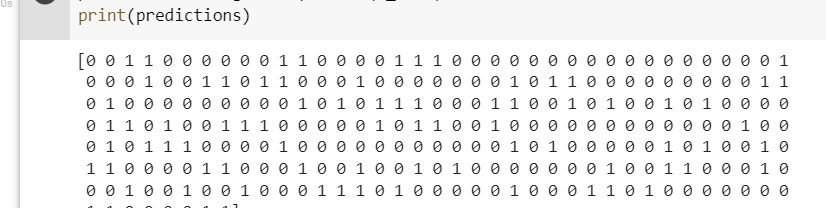
**logmodel = LogisticRegression(class\_weight = 'balanced')**

**# class\_weight setting balanced is powerful when there are more regrets than acceptance**

**logmodel.fit(X\_train,y\_train)**

**predictions\_log = logmodel.predict(X\_test)**

****

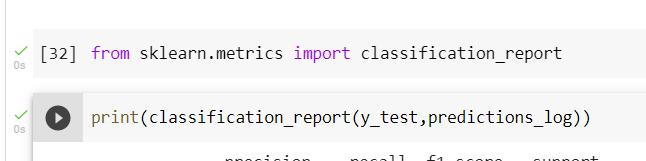


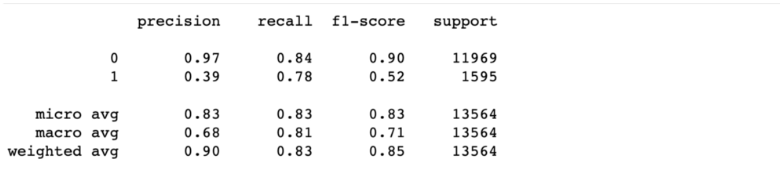
An argument class\_weight = balanced was passed to the model. This is just because I noticed in my very first data visualization that the number of High job incomes was more than the low ones.

In the form of yes and no.

Now making the classification report of our model:

**print(classification\_report(y\_test,predictions\_log))**

****

****

We can see that our model has a 97% precision rate for low Income jobs and a 39% precision rate for High income jobs. This is a result of our data imbalance.

**BUIDING CONFUSION MATRIX:**

**confusion\_matrix(y\_test,predictions\_log)**

array([[230 , 13],

[48 , 72]])

**THE END**