Configuration Manual

Introduction

This handbook contains information on the steps involved in carrying out the research's implementation. The handbook includes system settings, environment setup, relevant code snippets, and other information needed to reproduce this study.

Hardware Specification

The following are the hardware specifications of the system. A screenshot of the same is also provided.

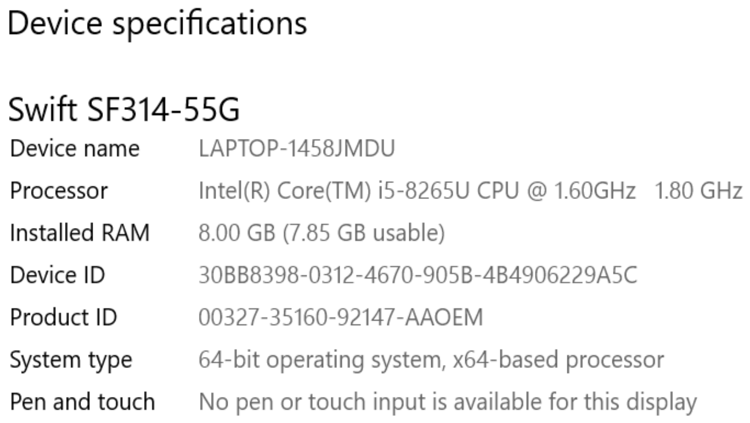
Host Device: Acer Swift SF314-55G

Processor: 1.80GHz Intel Core i5- 8265U

Memory: 8gb

Storage: 512GB SSD

Graphics: NVIDIA MX230



Device specifications

Swift SF314-55G

Device name	LAPTOP-1458JMDU
Processor	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
Installed RAM	8.00 GB (7.85 GB usable)
Device ID	30BB8398-0312-4670-905B-4B4906229A5C
Product ID	00327-35160-92147-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Software Specifications

All the programming is done in python in Jupyter notebook. Jupyter notebook is an open-source IDE for creating Jupyter documents, which may be written and shared with live code. It is also an interactive computational environment that is accessible over the web. The Jupyter notebook may handle a variety of data science languages, including Python, Julia, Scala, R, and others. The manual work on dataset was done using Microsoft Excel 365.

Data Source

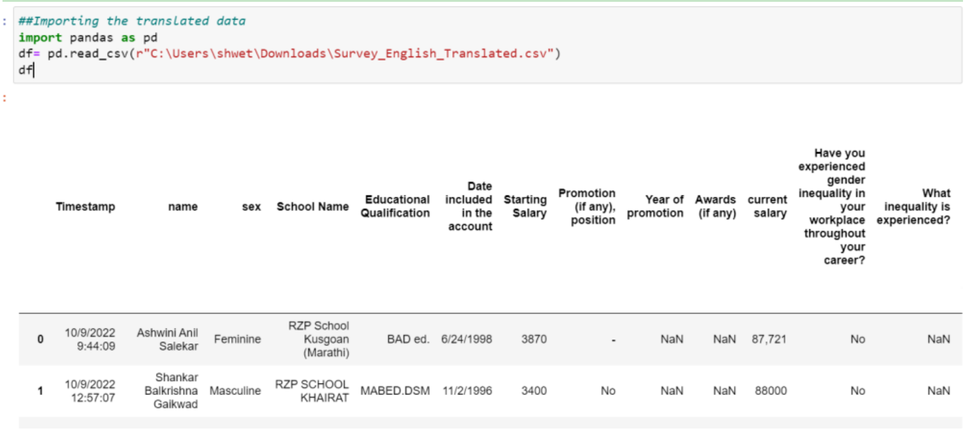
The data is collected through empirical methods. A survey was conducted online by sharing a google form through WhatsApp groups. Two set of surveys were conducted, the second was based on the first one.

Original Dataset screenshot

Translate Dataset screenshot

Data Preprocessing and transformation

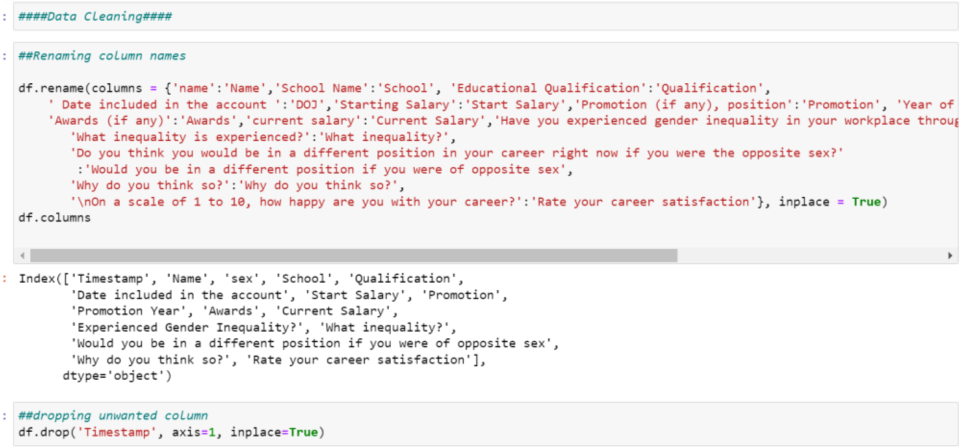
The data preprocessing and transformation was done programmatically



```
##Importing the translated data
import pandas as pd
df= pd.read_csv(r"C:\Users\shwet\Downloads\Survey_English_Translated.csv")
df
```

	Timestamp	name	sex	School Name	Educational Qualification	Date included in the account	Starting Salary	Promotion (if any), position	Year of promotion	Awards (if any)	current salary	Have you experienced gender inequality in your workplace throughout your career?	What inequality is experienced?
0	10/9/2022 9:44:09	Ashwini Anil Salekar	Feminine	RZP School Kusgoan (Marathi)	BAD ed.	6/24/1998	3870	-	NaN	NaN	87,721	No	NaN
1	10/9/2022 12:57:07	Shankar Balkrishna Gaikwad	Masculine	RZP SCHOOL KHAIRAT	MABED.DSM	11/2/1996	3400	No	NaN	NaN	88000	No	NaN

Importing data



```
####Data Cleaning####
```

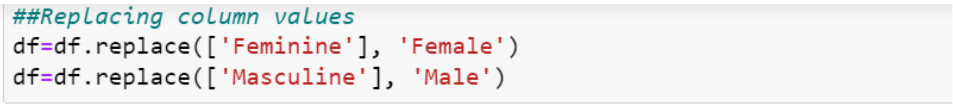
```
##Renaming column names

df.rename(columns = {'name':'Name','School Name':'School', 'Educational Qualification':'Qualification',
    ' Date included in the account ':'DOJ','Starting Salary':'Start Salary','Promotion (if any), position':'Promotion', 'Year of
    'Awards (if any)':'Awards','current salary':'Current Salary','Have you experienced gender inequality in your workplace throug
       'What inequality is experienced?':'What inequality?',
       'Do you think you would be in a different position in your career right now if you were the opposite sex?'
        :'Would you be in a different position if you were of opposite sex',
       'Why do you think so?':'Why do you think so?',
       '\nOn a scale of 1 to 10, how happy are you with your career?':'Rate your career satisfaction'}, inplace = True)
df.columns
```

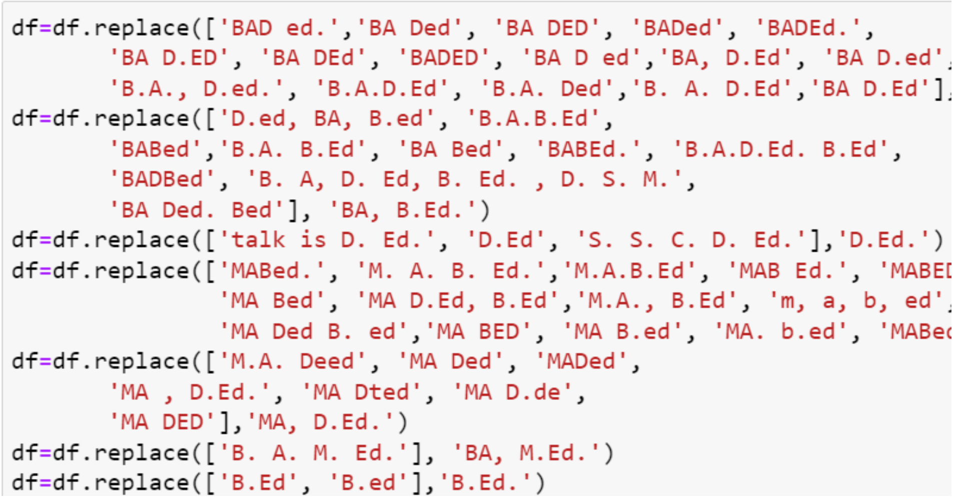
```
Index(['Timestamp', 'Name', 'sex', 'School', 'Qualification',
       'Date included in the account', 'Start Salary', 'Promotion',
       'Promotion Year', 'Awards', 'Current Salary',
       'Experienced Gender Inequality?', 'What inequality?',
       'Would you be in a different position if you were of opposite sex',
       'Why do you think so?', 'Rate your career satisfaction'],
      dtype='object')
```

```
##dropping unwanted column
df.drop('Timestamp', axis=1, inplace=True)
```

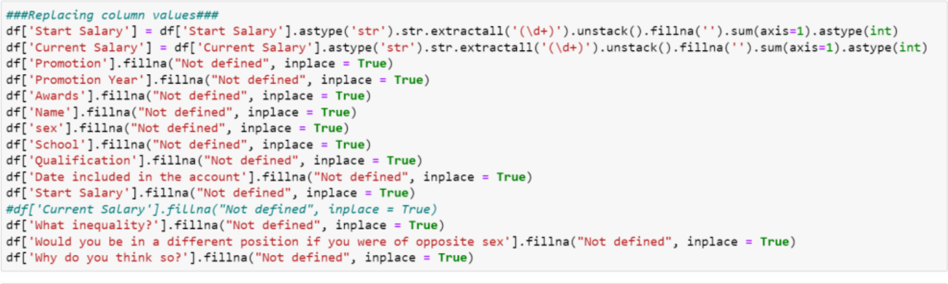
Renaming and dropping columns



```
##Replacing column values
df=df.replace(['Feminine'], 'Female')
df=df.replace(['Masculine'], 'Male')
```

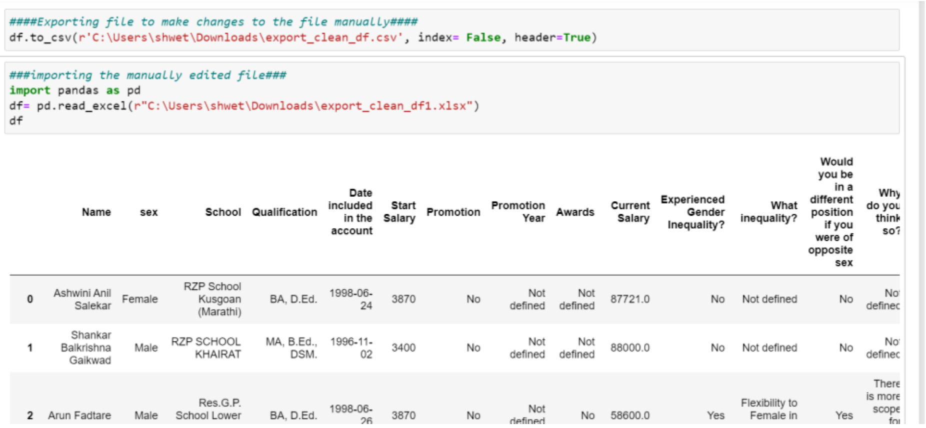


```
df=df.replace(['BAD ed.','BA Ded', 'BA DED', 'BADed', 'BADEd.',
       'BA D.ED', 'BA DEd', 'BADED', 'BA D ed','BA, D.Ed', 'BA D.ed',
       'B.A., D.ed.', 'B.A.D.Ed', 'B.A. Ded','B. A. D.Ed','BA D.Ed'],
df=df.replace(['D.ed, BA, B.ed', 'B.A.B.Ed',
       'BABed','B.A. B.Ed', 'BA Bed', 'BABEd.', 'B.A.D.Ed. B.Ed',
       'BADBed', 'B. A, D. Ed, B. Ed. , D. S. M.',
       'BA Ded. Bed'], 'BA, B.Ed.')
df=df.replace(['talk is D. Ed.', 'D.Ed', 'S. S. C. D. Ed.'],'D.Ed.')
df=df.replace(['MABed.', 'M. A. B. Ed.','M.A.B.Ed', 'MAB Ed.', 'MABEI
               'MA Bed', 'MA D.Ed, B.Ed','M.A., B.Ed', 'm, a, b, ed',
               'MA Ded B. ed','MA BED', 'MA B.ed', 'MA. b.ed', 'MABe
df=df.replace(['M.A. Deed', 'MA Ded', 'MADed',
       'MA , D.Ed.', 'MA Dted', 'MA D.de',
       'MA DED'],'MA, D.Ed.')
df=df.replace(['B. A. M. Ed.'], 'BA, M.Ed.')
df=df.replace(['B.Ed', 'B.ed'],'B.Ed.')
```



```
###Replacing column values###
df['Start Salary'] = df['Start Salary'].astype('str').str.extractall('(\d+)').unstack().fillna('').sum(axis=1).astype(int)
df['Current Salary'] = df['Current Salary'].astype('str').str.extractall('(\d+)').unstack().fillna('').sum(axis=1).astype(int)
df['Promotion'].fillna("Not defined", inplace = True)
df['Promotion Year'].fillna("Not defined", inplace = True)
df['Awards'].fillna("Not defined", inplace = True)
df['Name'].fillna("Not defined", inplace = True)
df['sex'].fillna("Not defined", inplace = True)
df['School'].fillna("Not defined", inplace = True)
df['Qualification'].fillna("Not defined", inplace = True)
df['Date included in the account'].fillna("Not defined", inplace = True)
df['Start Salary'].fillna("Not defined", inplace = True)
#df['Current Salary'].fillna("Not defined", inplace = True)
df['What inequality?'].fillna("Not defined", inplace = True)
df['Would you be in a different position if you were of opposite sex'].fillna("Not defined", inplace = True)
df['Why do you think so?'].fillna("Not defined", inplace = True)
```

Replacing column values

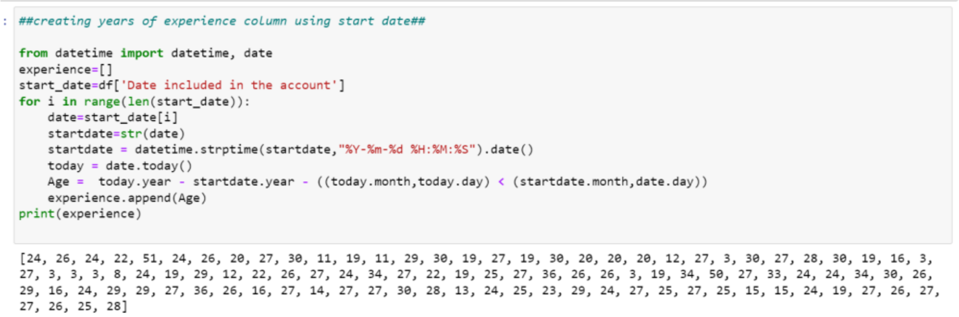


```
####Exporting file to make changes to the file manually####
df.to_csv(r'C:\Users\shwet\Downloads\export_clean_df.csv', index= False, header=True)
```

```
###importing the manually edited file###
import pandas as pd
df= pd.read_excel(r"C:\Users\shwet\Downloads\export_clean_df1.xlsx")
df
```

	Name	sex	School	Qualification	Date included in the account	Start Salary	Promotion	Promotion Year	Awards	Current Salary	Experienced Gender Inequality?	What inequality?	Would you be in a different position if you were of opposite sex	Why do you think so?
0	Ashwini Anil Salekar	Female	RZP School Kusgoan (Marathi)	BA, D.Ed.	1998-06-24	3870	No	Not defined	Not defined	87721.0	No	Not defined	No	Not defined
1	Shankar Balkrishna Gaikwad	Male	RZP SCHOOL KHAIRAT	MA, B.Ed., DSM.	1996-11-02	3400	No	Not defined	Not defined	88000.0	No	Not defined	No	Not defined
2	Arun Fadtare	Male	Res.G.P. School Lower	BA, D.Ed.	1998-06-26	3870	No	Not defined	No	58600.0	Yes	Flexibility to Female in	Yes	There is more scope for

Importing manually edited file

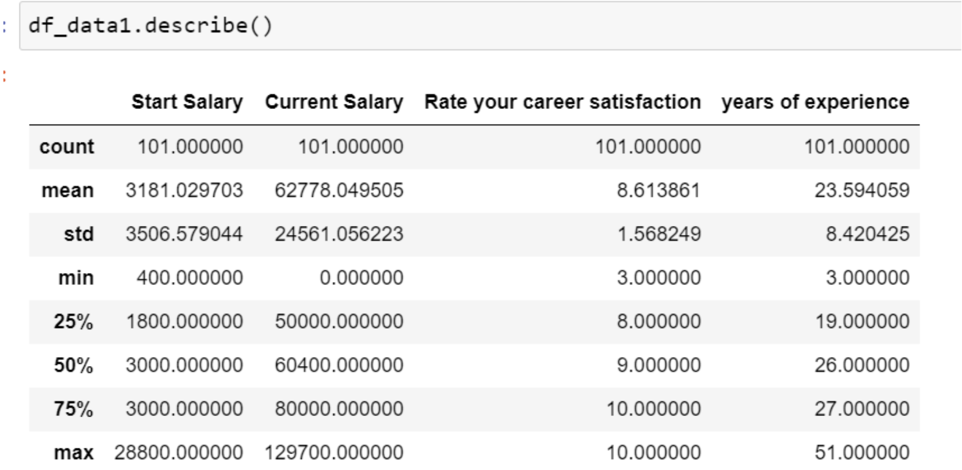


```
##creating years of experience column using start date##

from datetime import datetime, date
experience=[]
start_date=df['Date included in the account']
for i in range(len(start_date)):
    date=start_date[i]
    startdate=str(date)
    startdate = datetime.strptime(startdate,"%Y-%m-%d %H:%M:%S").date()
    today = date.today()
    Age =  today.year - startdate.year - ((today.month,today.day) < (startdate.month,date.day))
    experience.append(Age)
print(experience)
```

```
[24, 26, 24, 22, 51, 24, 26, 20, 27, 30, 11, 19, 11, 29, 30, 19, 27, 19, 30, 20, 20, 20, 12, 27, 3, 30, 27, 28, 30, 19, 16, 3,
27, 3, 3, 3, 8, 24, 19, 29, 12, 22, 26, 27, 24, 34, 27, 22, 19, 25, 27, 36, 26, 26, 3, 19, 34, 50, 27, 33, 24, 24, 34, 30, 26,
29, 16, 24, 29, 29, 27, 36, 26, 16, 27, 14, 27, 27, 30, 28, 13, 24, 25, 23, 29, 24, 27, 25, 27, 25, 15, 15, 24, 19, 27, 26, 27,
27, 26, 25, 28]
```

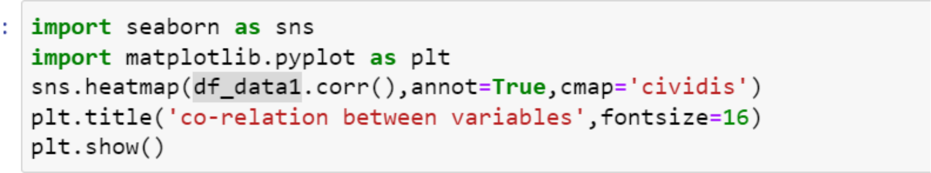
Creating new column



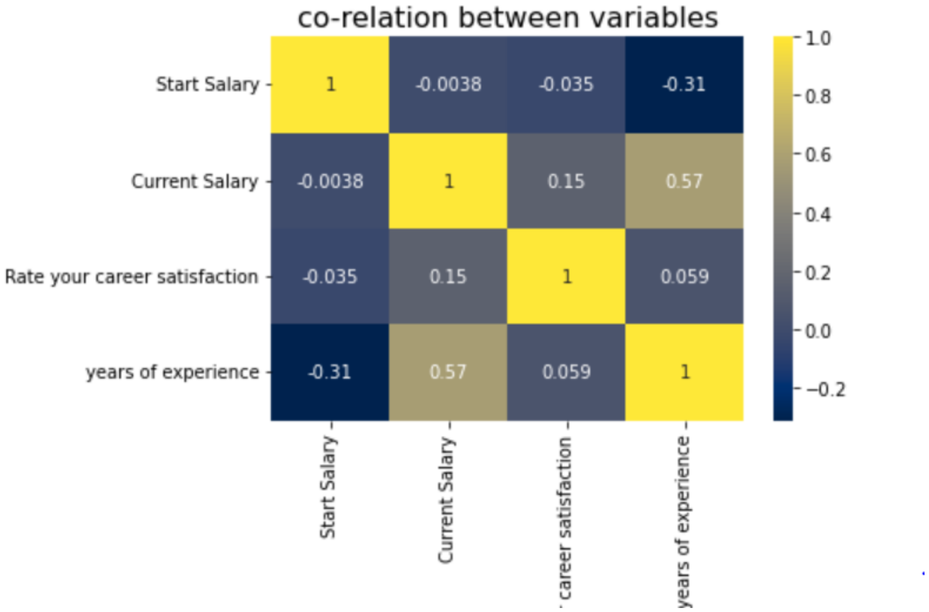
```
df_data1.describe()
```

	Start Salary	Current Salary	Rate your career satisfaction	years of experience
count	101.000000	101.000000	101.000000	101.000000
mean	3181.029703	62778.049505	8.613861	23.594059
std	3506.579044	24561.056223	1.568249	8.420425
min	400.000000	0.000000	3.000000	3.000000
25%	1800.000000	50000.000000	8.000000	19.000000
50%	3000.000000	60400.000000	9.000000	26.000000
75%	3000.000000	80000.000000	10.000000	27.000000
max	28800.000000	129700.000000	10.000000	51.000000

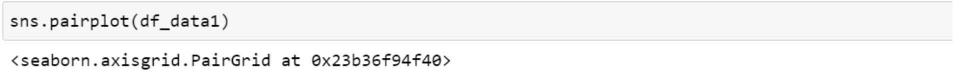
Data description



```
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(df_data1.corr(),annot=True,cmap='cividis')
plt.title('co-relation between variables',fontsize=16)
plt.show()
```

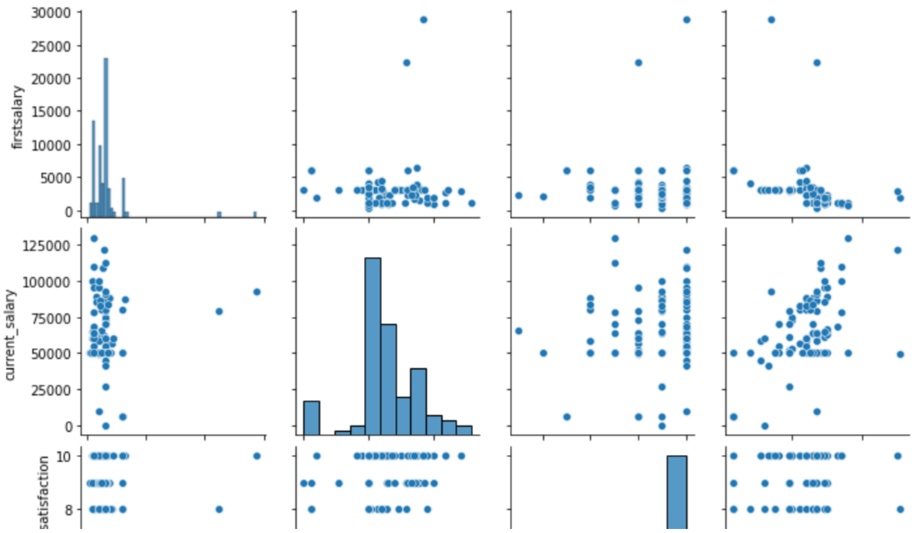


Confusion matrix



```
sns.pairplot(df_data1)
```

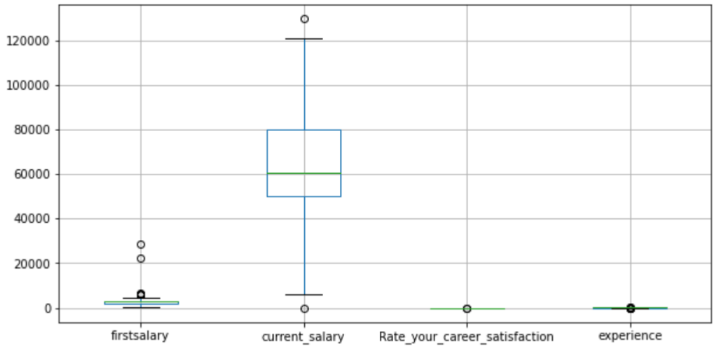
```
<seaborn.axisgrid.PairGrid at 0x23b36f94f40>
```



Pairplot

```
plt.figure(figsize=(10,5))
df_data1.boxplot(column=['firstsalary', 'current_salary', 'Rate_your_career_satisfaction','experience'])
```

```
<AxesSubplot:>
```

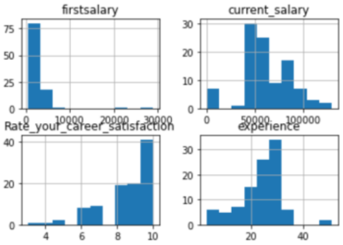


Boxplot

```
plt.figure(figsize=(100,100))
df_data1[['sex','Qualification','firstsalary', 'Promotion', 'Awards', 'current_salary', 'experienced_inequality', 'Rate_your_care
```

```
array([[<AxesSubplot:title={'center':'firstsalary'}>,
        <AxesSubplot:title={'center':'current_salary'}>],
       [<AxesSubplot:title={'center':'Rate_your_career_satisfaction'}>,
        <AxesSubplot:title={'center':'experience'}>]], dtype=object)

<Figure size 7200x7200 with 0 Axes>
```



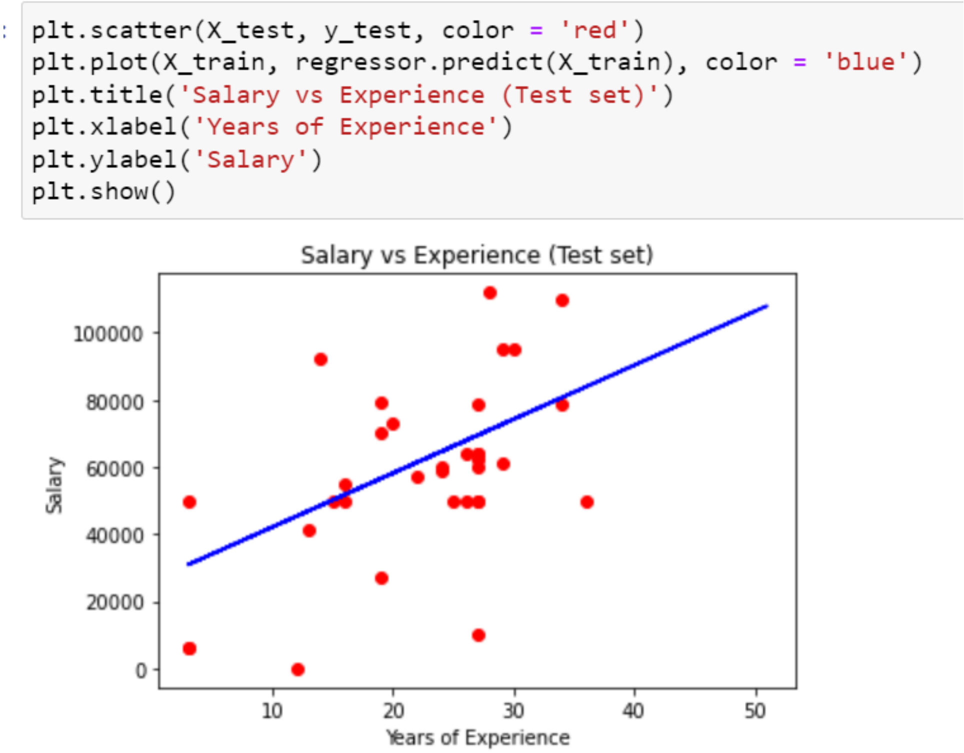
Histograms

```
##Linear Regression

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
X = df_newdata1.iloc[:, :-1].values
y = df_newdata1.iloc[:, -1].values
```

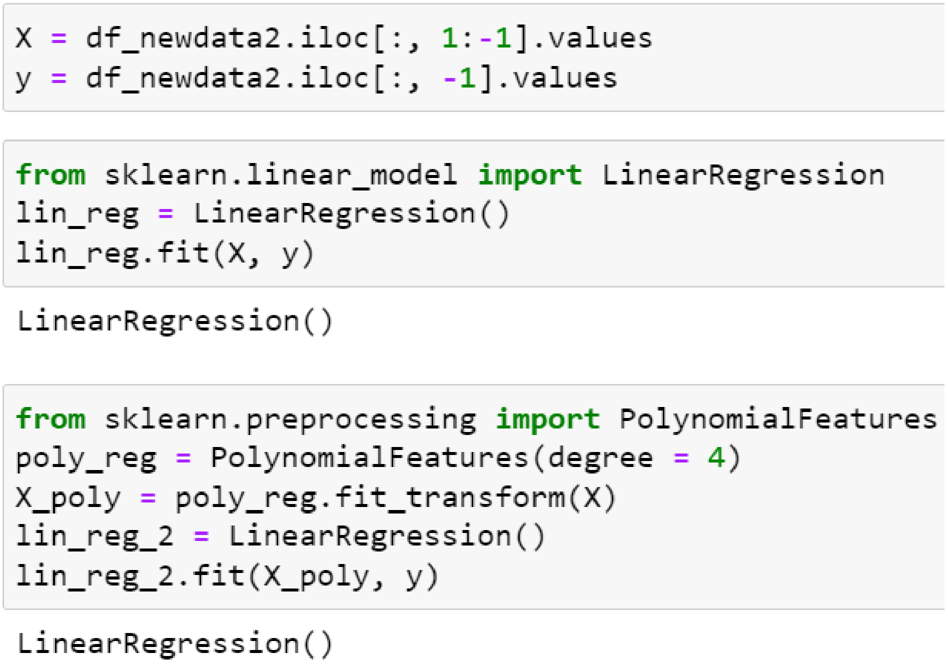
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)
X_train= X_train.reshape(-1, 1)
X_test = X_test.reshape(-1, 1)
y_train
```

```
array([129700.,  50000.,  63000., 100000.,  50000.,  91090.,  88000.,
        60400.,  70000.,  80000.,  49500.,  61000.,  50000.,  60000.,
        50000.,  87000.,  83000.,  68000.,  87721.,   6000.,  66100.,
        58000.,  52000.,   6000.,  61600.,  64000.,  45000.,   6000.,
```



```
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

Linear regression



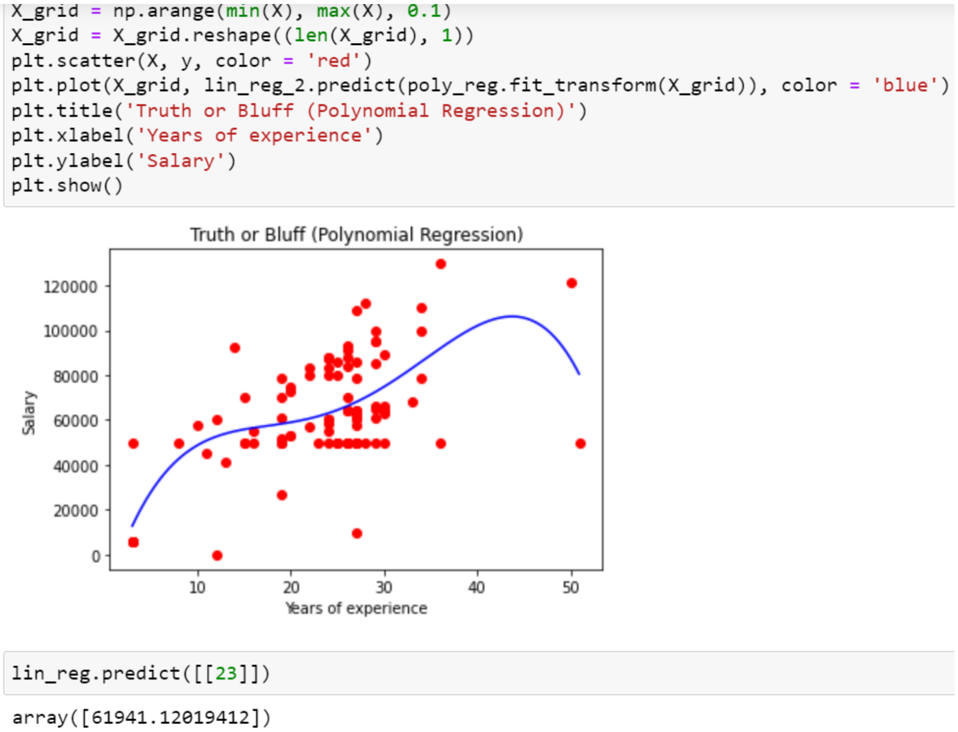
```
X = df_newdata2.iloc[:, 1:-1].values
y = df_newdata2.iloc[:, -1].values
```

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)
```

```
LinearRegression()
```

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4)
X_poly = poly_reg.fit_transform(X)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)
```

```
LinearRegression()
```

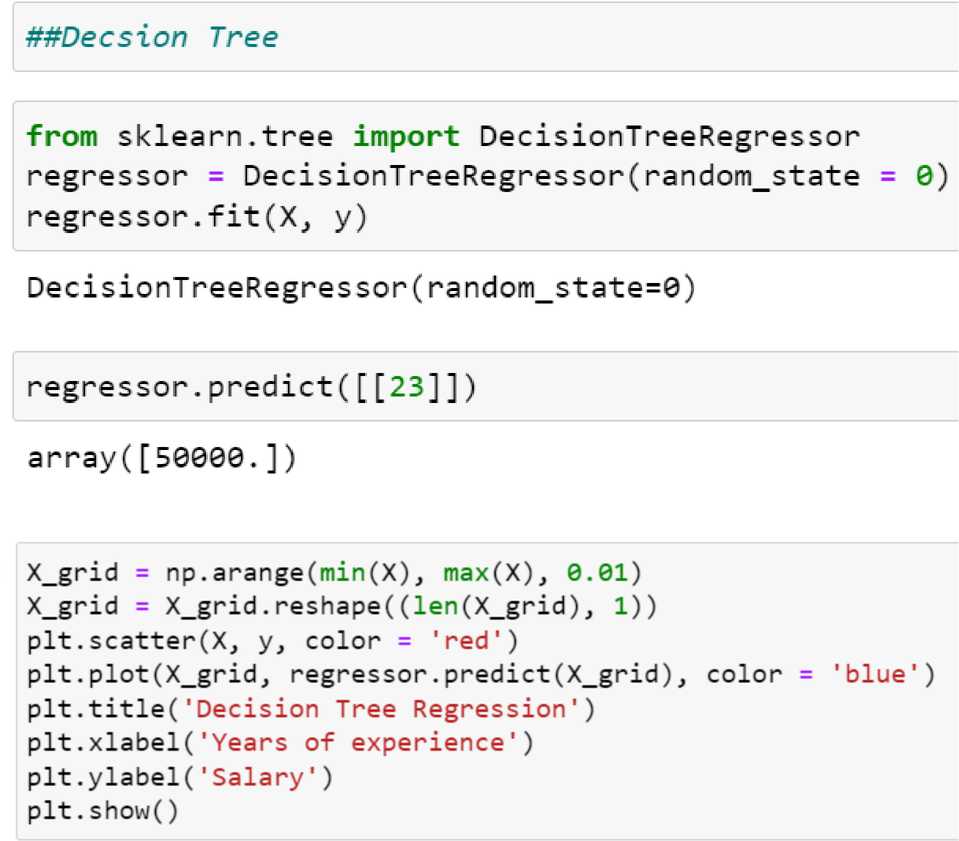


```
X_grid = np.arange(min(X), max(X), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, lin_reg_2.predict(poly_reg.fit_transform(X_grid)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Years of experience')
plt.ylabel('Salary')
plt.show()
```

```
lin_reg.predict([[23]])
```

```
array([61941.12019412])
```

Polynomial Regression



```
##Decsion Tree
```

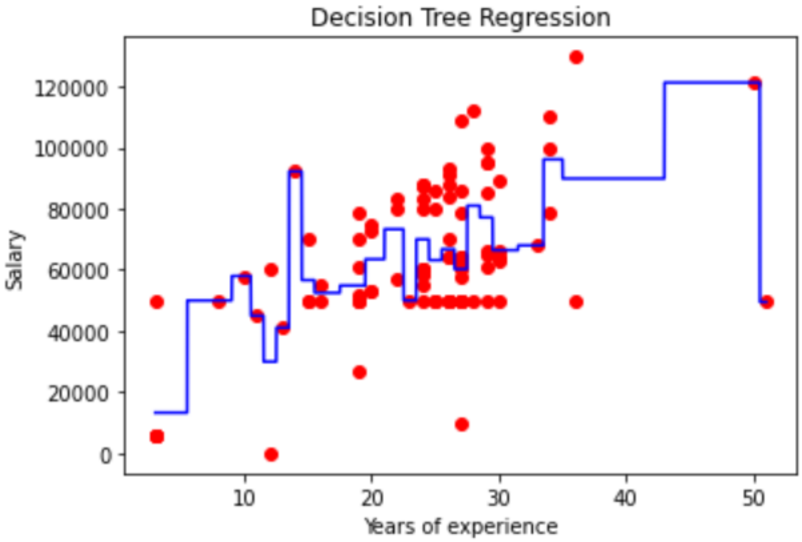
```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X, y)
```

```
DecisionTreeRegressor(random_state=0)
```

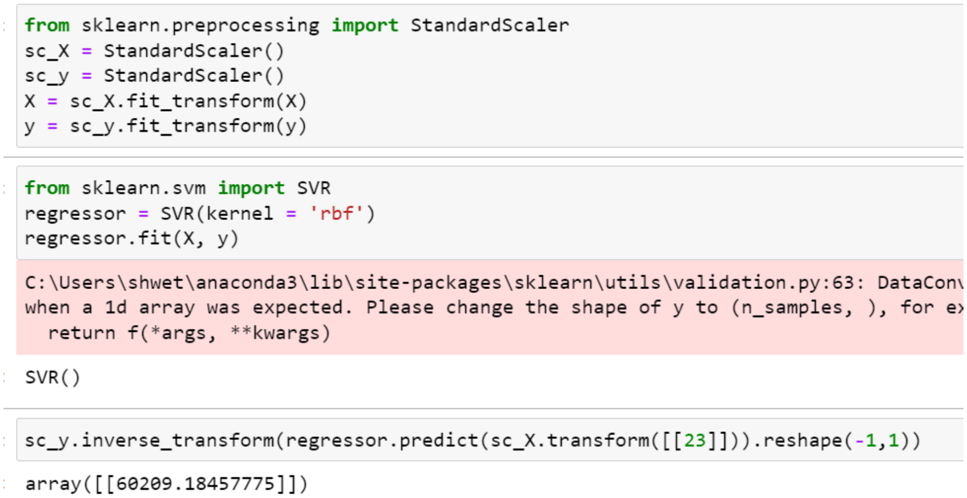
```
regressor.predict([[23]])
```

```
array([50000.])
```

```
X_grid = np.arange(min(X), max(X), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.title('Decision Tree Regression')
plt.xlabel('Years of experience')
plt.ylabel('Salary')
plt.show()
```



Decision Tree Regression



```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X)
y = sc_y.fit_transform(y)
```

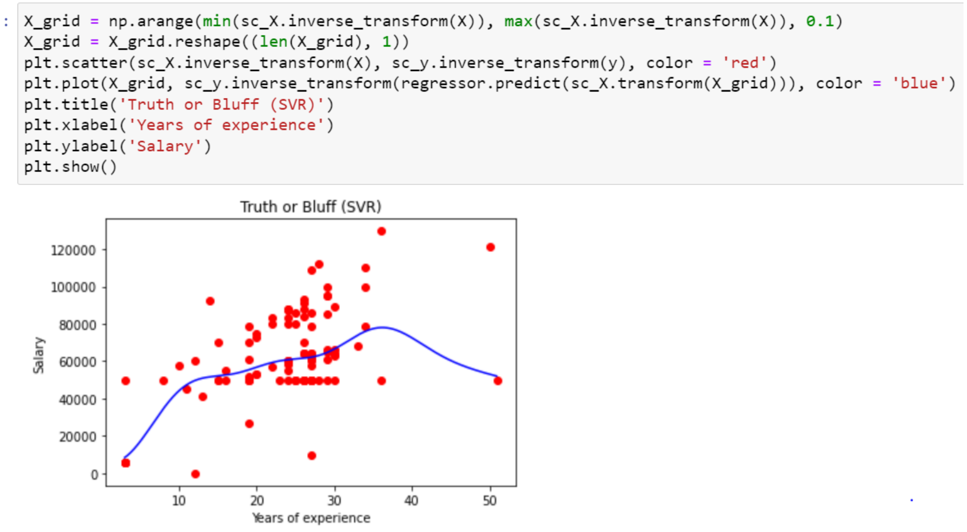
```
from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf')
regressor.fit(X, y)
```

```
C:\Users\shwet\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConv
when a 1d array was expected. Please change the shape of y to (n_samples, ), for ex
  return f(*args, **kwargs)
```

```
SVR()
```

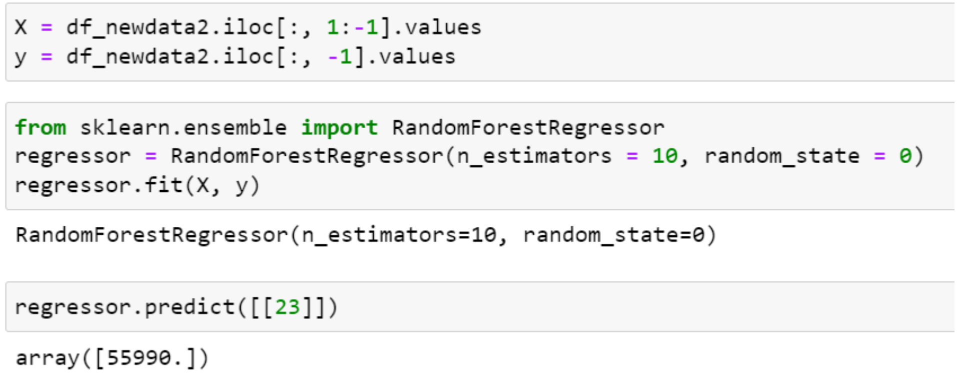
```
sc_y.inverse_transform(regressor.predict(sc_X.transform([[23]])).reshape(-1,1))
```

```
array([[60209.18457775]])
```



```
X_grid = np.arange(min(sc_X.inverse_transform(X)), max(sc_X.inverse_transform(X)), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(sc_X.inverse_transform(X), sc_y.inverse_transform(y), color = 'red')
plt.plot(X_grid, sc_y.inverse_transform(regressor.predict(sc_X.transform(X_grid))), color = 'blue')
plt.title('Truth or Bluff (SVR)')
plt.xlabel('Years of experience')
plt.ylabel('Salary')
plt.show()
```

SVR



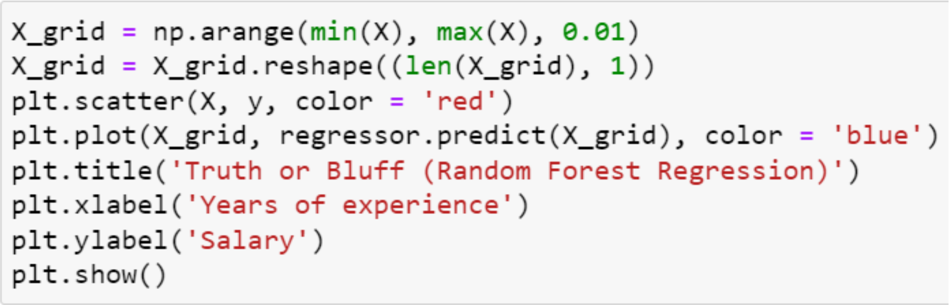
```
X = df_newdata2.iloc[:, 1:-1].values
y = df_newdata2.iloc[:, -1].values
```

```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(X, y)
```

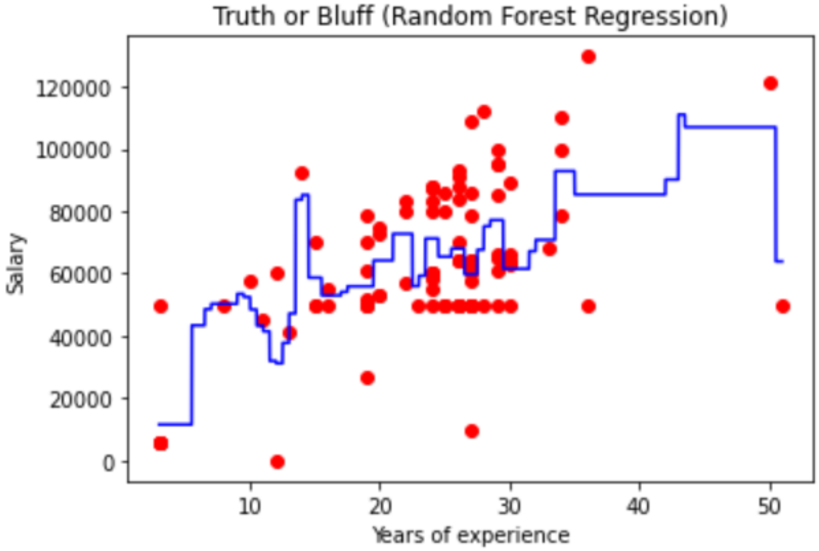
```
RandomForestRegressor(n_estimators=10, random_state=0)
```

```
regressor.predict([[23]])
```

```
array([55990.])
```

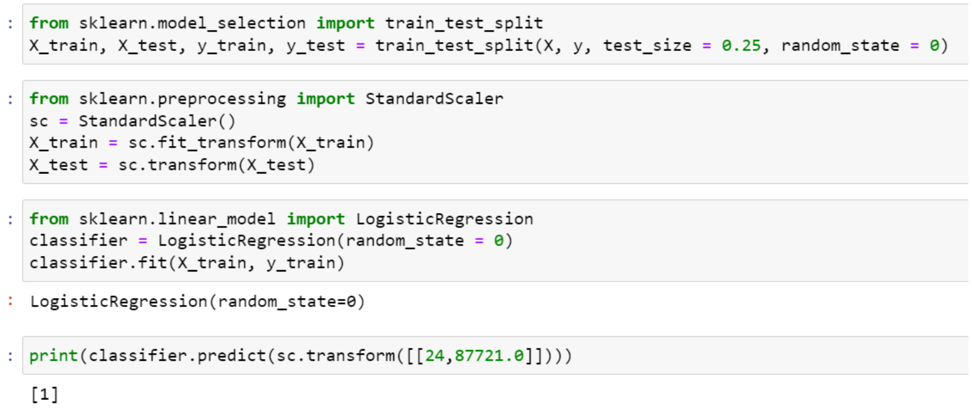


```
X_grid = np.arange(min(X), max(X), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.title('Truth or Bluff (Random Forest Regression)')
plt.xlabel('Years of experience')
plt.ylabel('Salary')
plt.show()
```



Random Forest

CLASSIFICATION



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

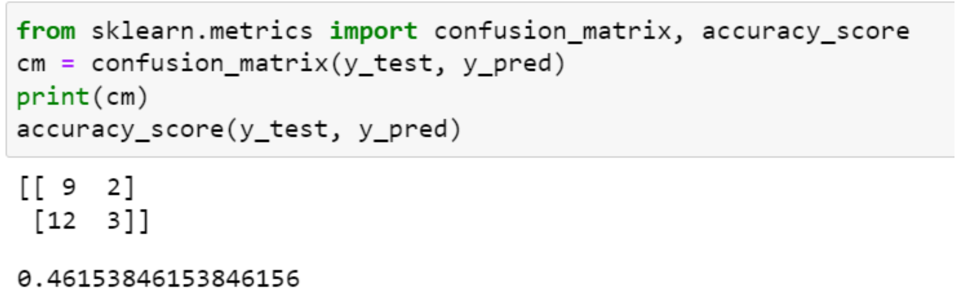
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

```
LogisticRegression(random_state=0)
```

```
print(classifier.predict(sc.transform([[24,87721.0]])))
```

```
[1]
```

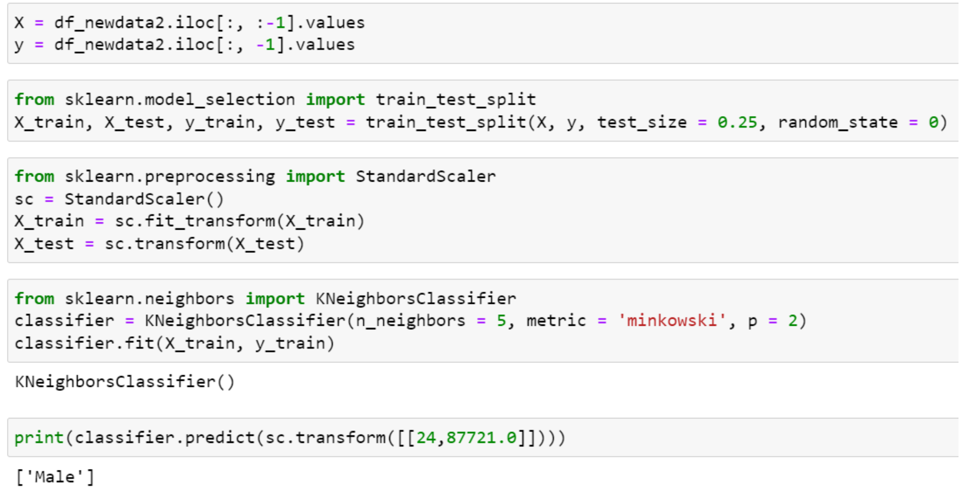


```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[ 9  2]
 [12  3]]

0.46153846153846156
```

Logistic Regression



```
X = df_newdata2.iloc[:, :-1].values
y = df_newdata2.iloc[:, -1].values
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

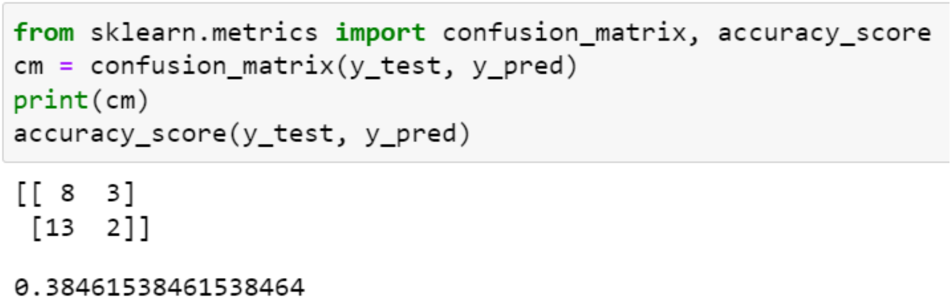
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
classifier.fit(X_train, y_train)
```

```
KNeighborsClassifier()
```

```
print(classifier.predict(sc.transform([[24,87721.0]])))
```

```
['Male']
```

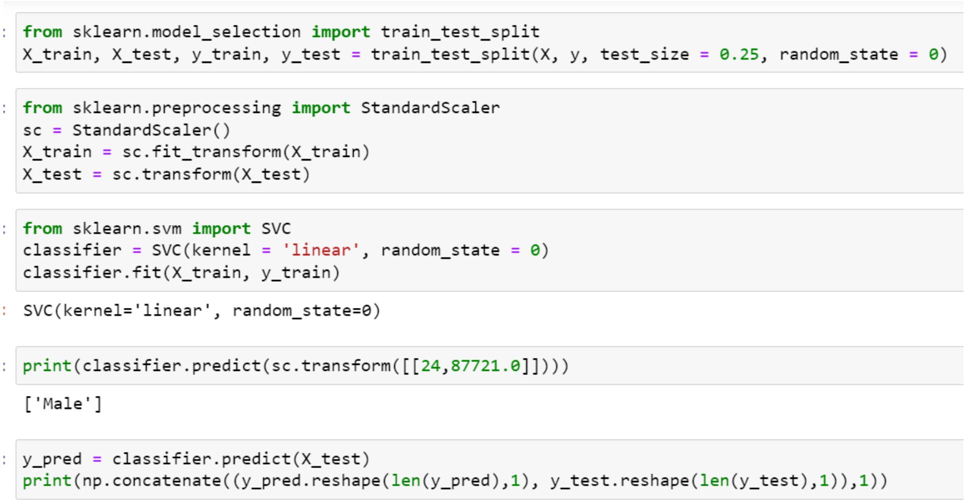


```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[ 8  3]
 [13  2]]

0.38461538461538464
```

KNN



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

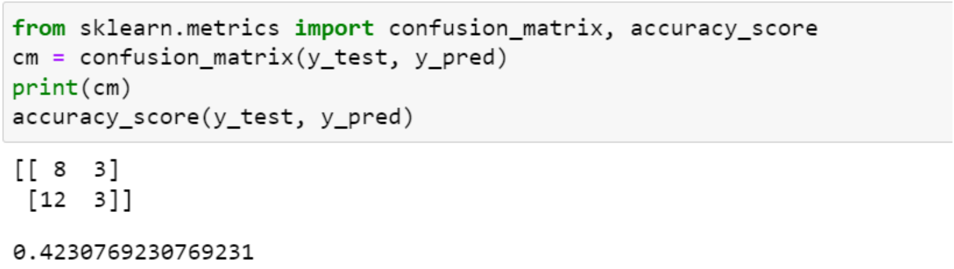
```
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)
```

```
SVC(kernel='linear', random_state=0)
```

```
print(classifier.predict(sc.transform([[24,87721.0]])))
```

```
['Male']
```

```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

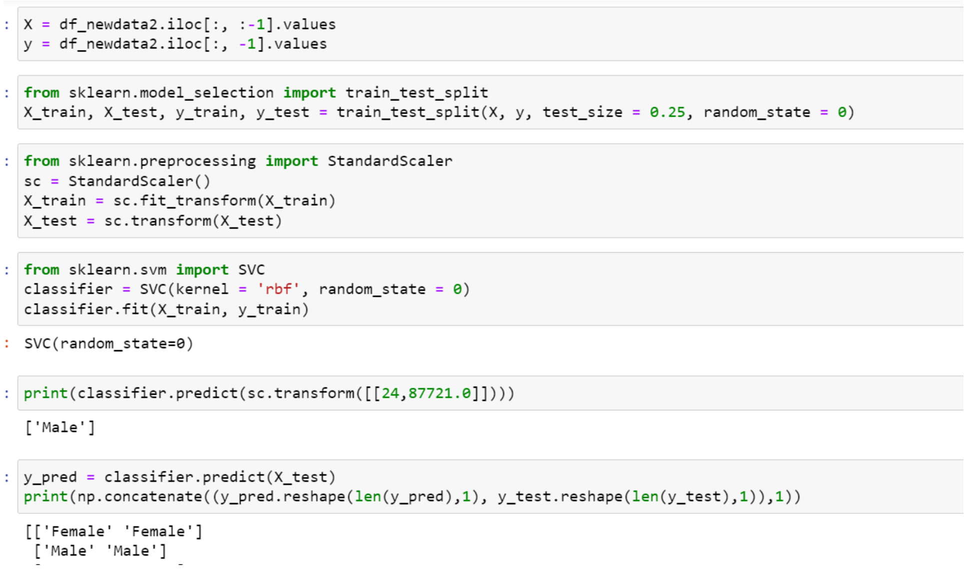


```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[ 8  3]
 [12  3]]

0.4230769230769231
```

SVM



```
X = df_newdata2.iloc[:, :-1].values
y = df_newdata2.iloc[:, -1].values
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, y_train)
```

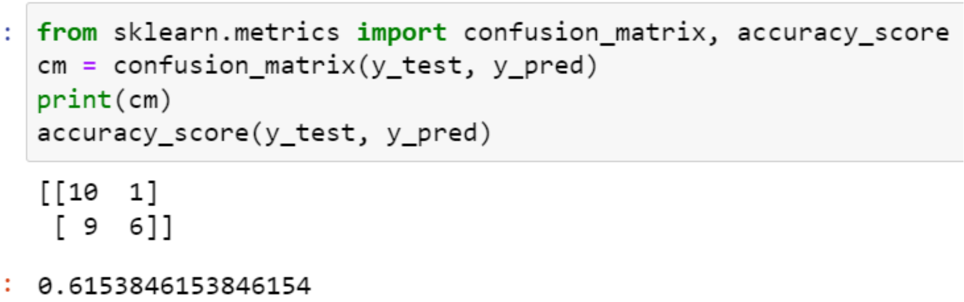
```
SVC(random_state=0)
```

```
print(classifier.predict(sc.transform([[24,87721.0]])))
```

```
['Male']
```

```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
[['Female' 'Female']
 ['Male' 'Male']
```

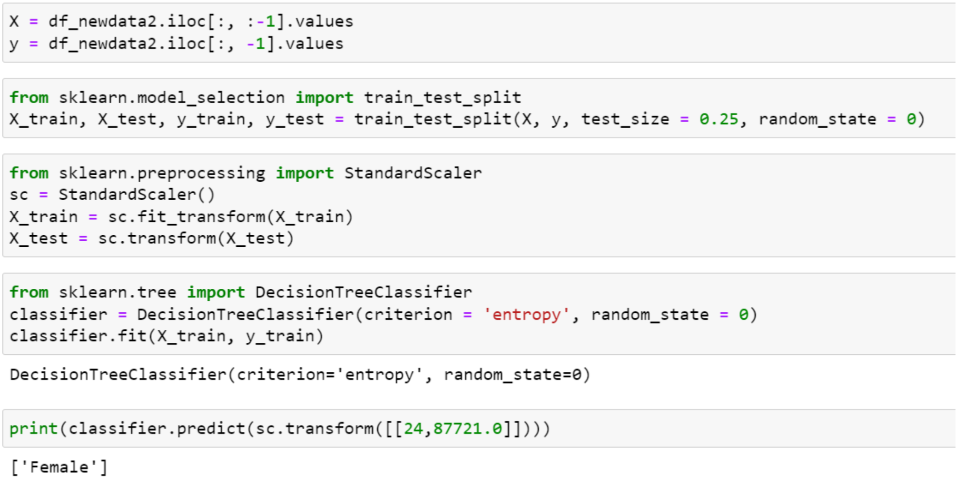


```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[10  1]
 [ 9  6]]
```

```
0.6153846153846154
```

Kernel SVM



```
X = df_newdata2.iloc[:, :-1].values
y = df_newdata2.iloc[:, -1].values
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

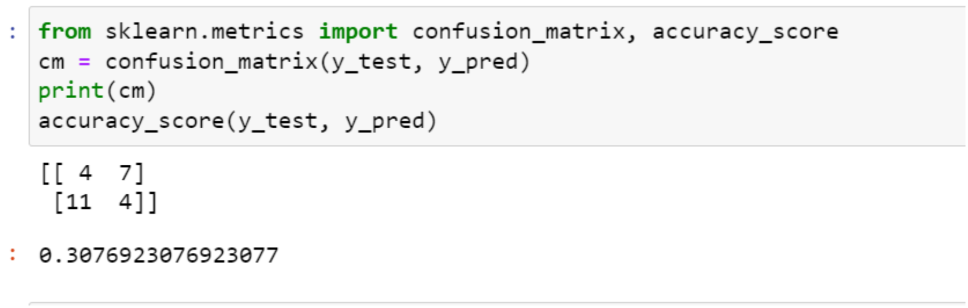
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

```
print(classifier.predict(sc.transform([[24,87721.0]])))
```

```
['Female']
```

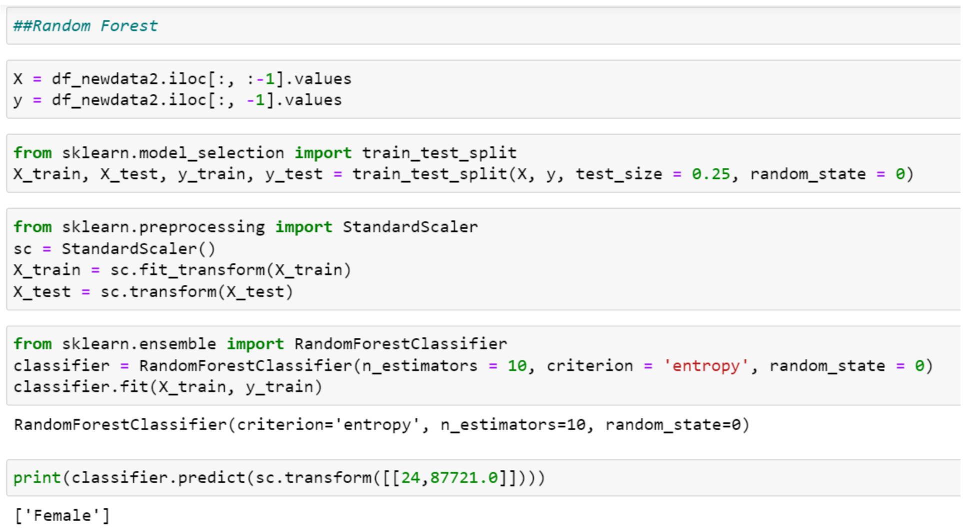


```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[ 4  7]
 [11  4]]
```

```
0.3076923076923077
```

Decision Tree Classifier



```
##Random Forest
```

```
X = df_newdata2.iloc[:, :-1].values
y = df_newdata2.iloc[:, -1].values
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

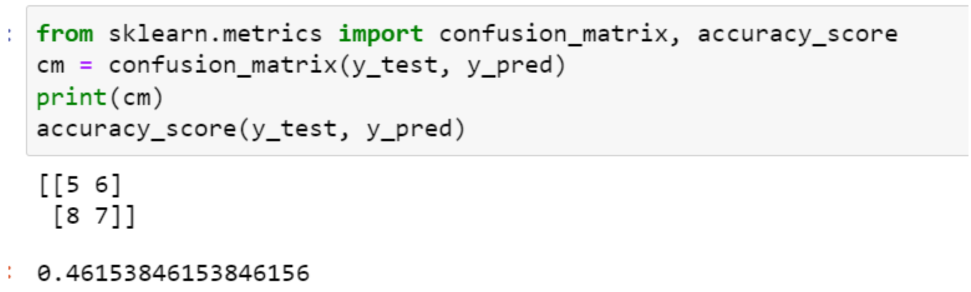
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)
```

```
print(classifier.predict(sc.transform([[24,87721.0]])))
```

```
['Female']
```



```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[5 6]
 [8 7]]
```

```
0.46153846153846156
```

Random Forest

DATA VISUALIZATION

Implementation