ROHAN NYATI

500075940

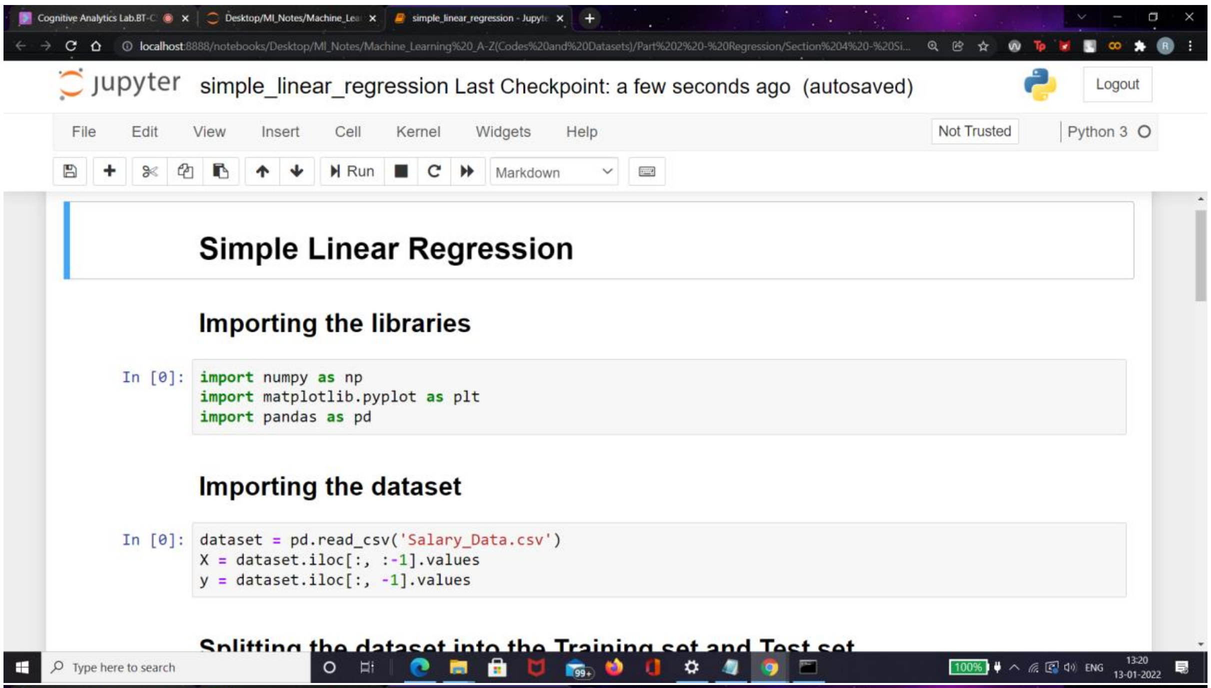
R177219148

BATCH-5 (Ai&Ml)

<u>Experiment – 1</u>

What is Regression?

Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables. More specifically, Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed. It predicts continuous/real values such as **temperature, age, salary, price,** etc.


jupyter simple_linear_regression Last Checkpoint: a few seconds ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown

Simple Linear Regression

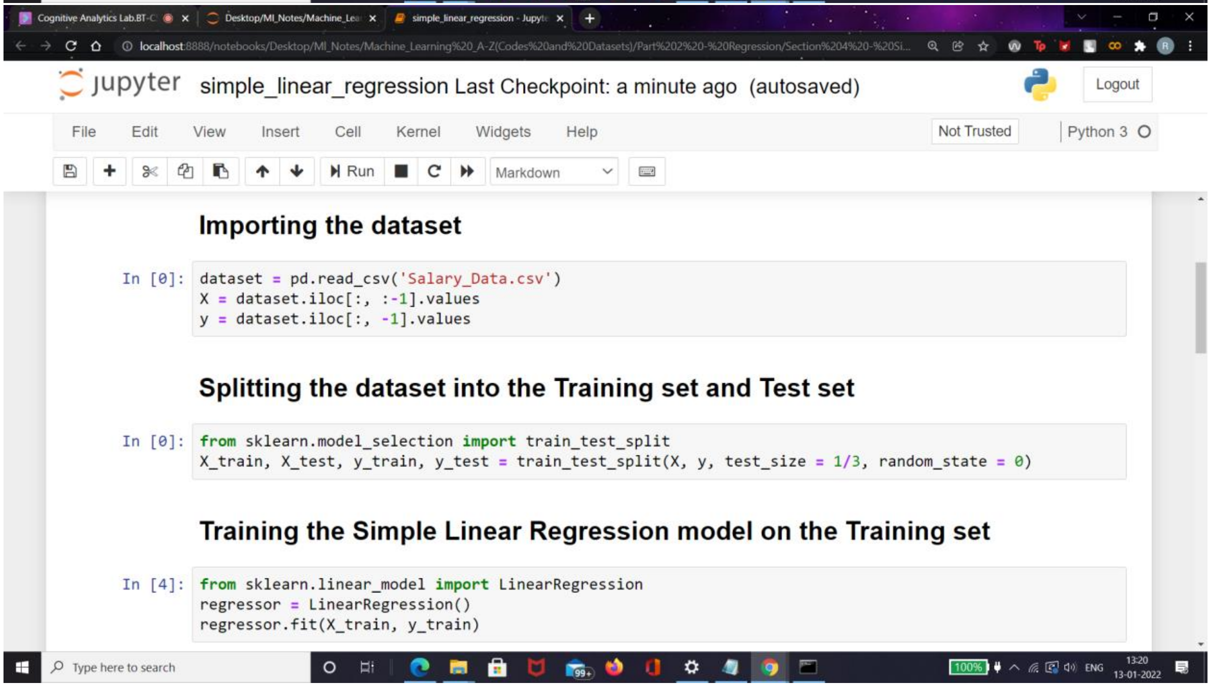
Importing the libraries

In [0]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

In [0]:
```
dataset = pd.read_csv('Salary_Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

Splitting the dataset into the Training set and Test set


jupyter simple_linear_regression Last Checkpoint: a minute ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown

Importing the dataset

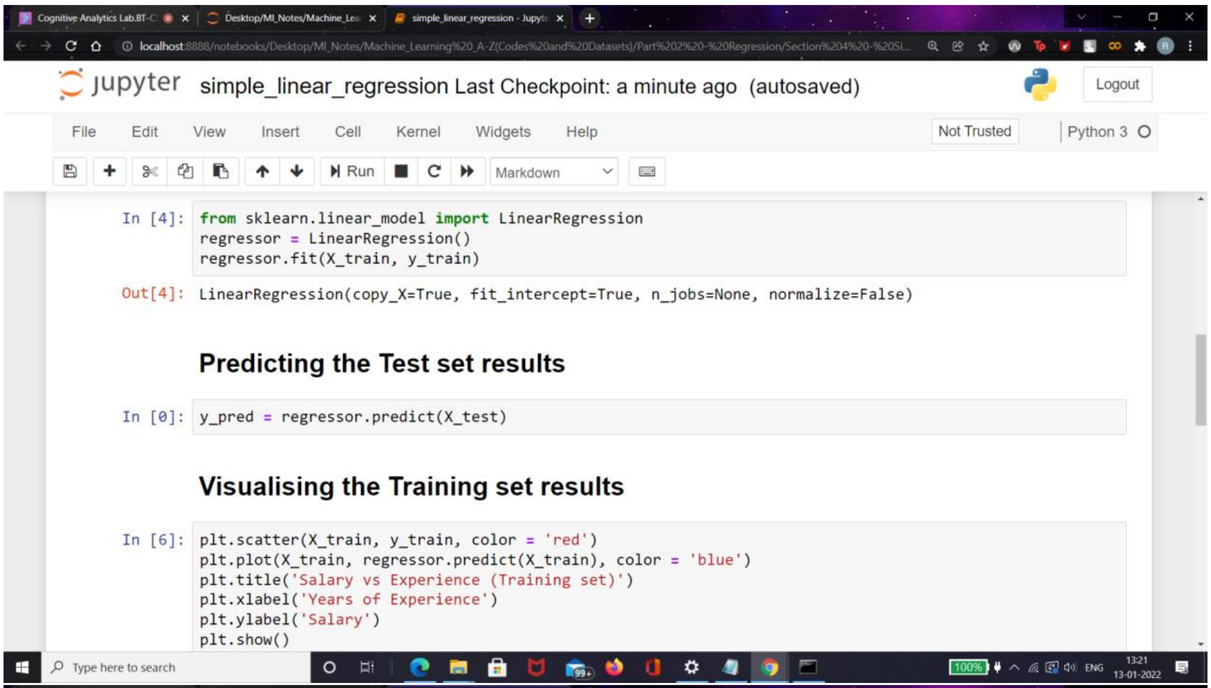
In [0]:
```
dataset = pd.read_csv('Salary_Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

Splitting the dataset into the Training set and Test set

In [0]:
```
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)
```

Training the Simple Linear Regression model on the Training set

In [4]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```



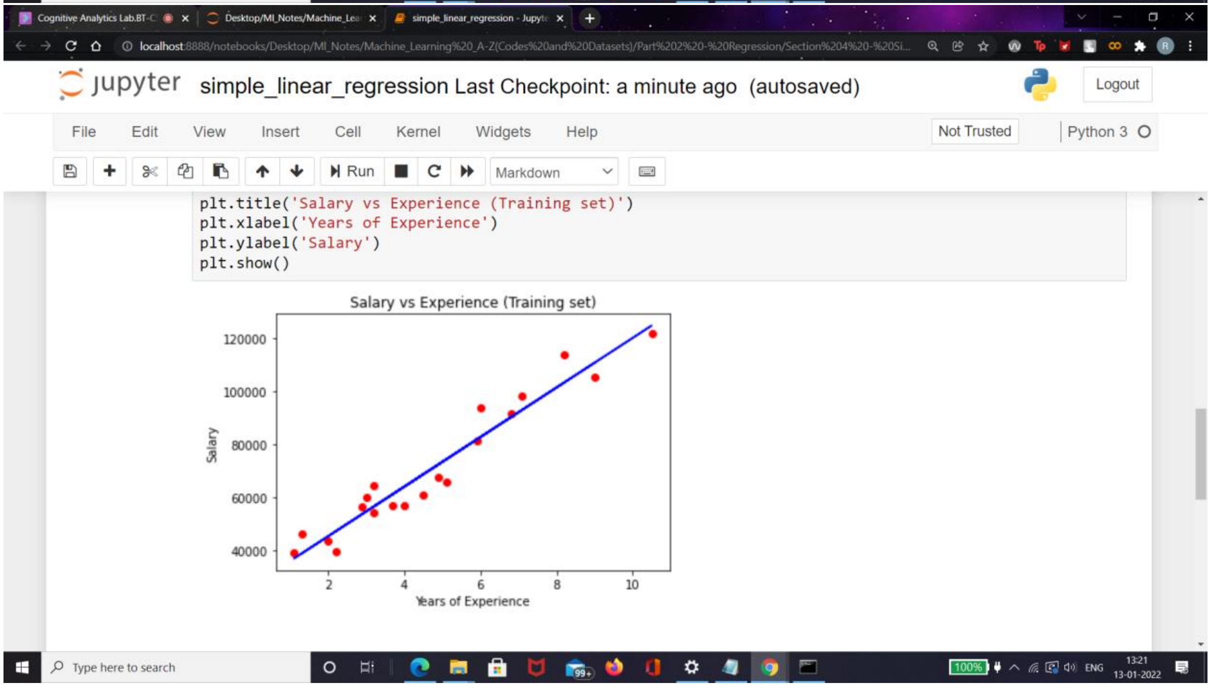
```
In [4]: from sklearn.linear_model import LinearRegression
        regressor = LinearRegression()
        regressor.fit(X_train, y_train)

Out[4]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Predicting the Test set results

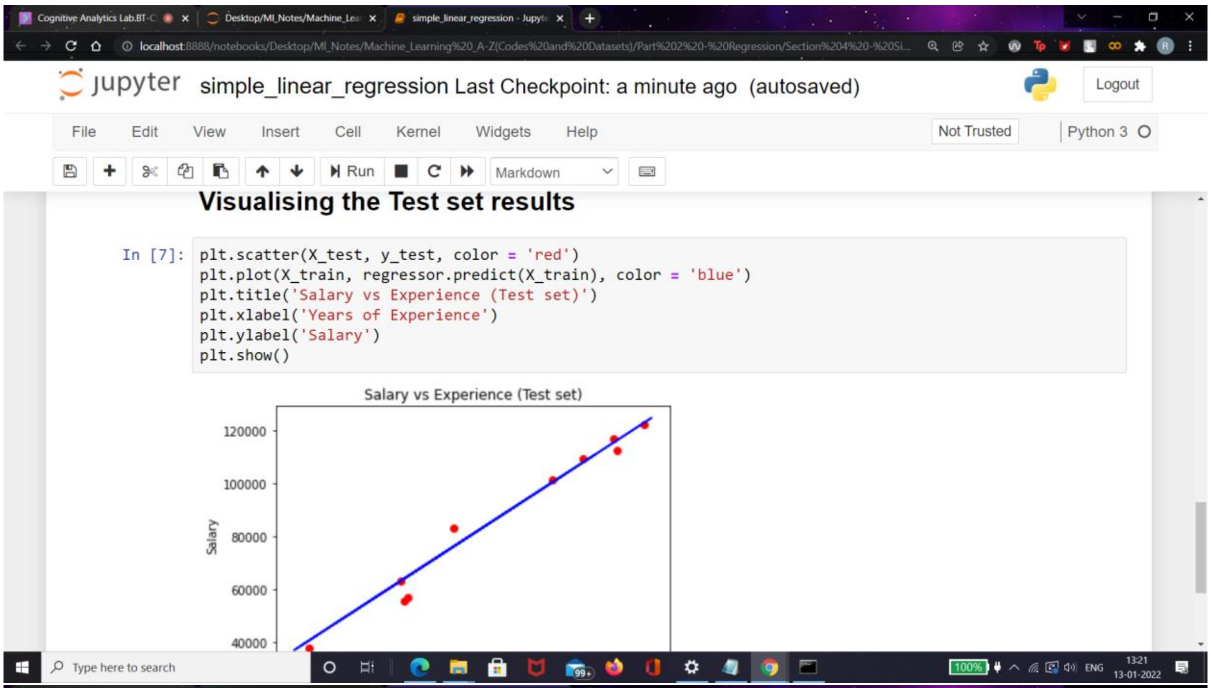
```
In [0]: y_pred = regressor.predict(X_test)
```

Visualising the Training set results

```
In [6]: plt.scatter(X_train, y_train, color = 'red')
        plt.plot(X_train, regressor.predict(X_train), color = 'blue')
        plt.title('Salary vs Experience (Training set)')
        plt.xlabel('Years of Experience')
        plt.ylabel('Salary')
        plt.show()
```



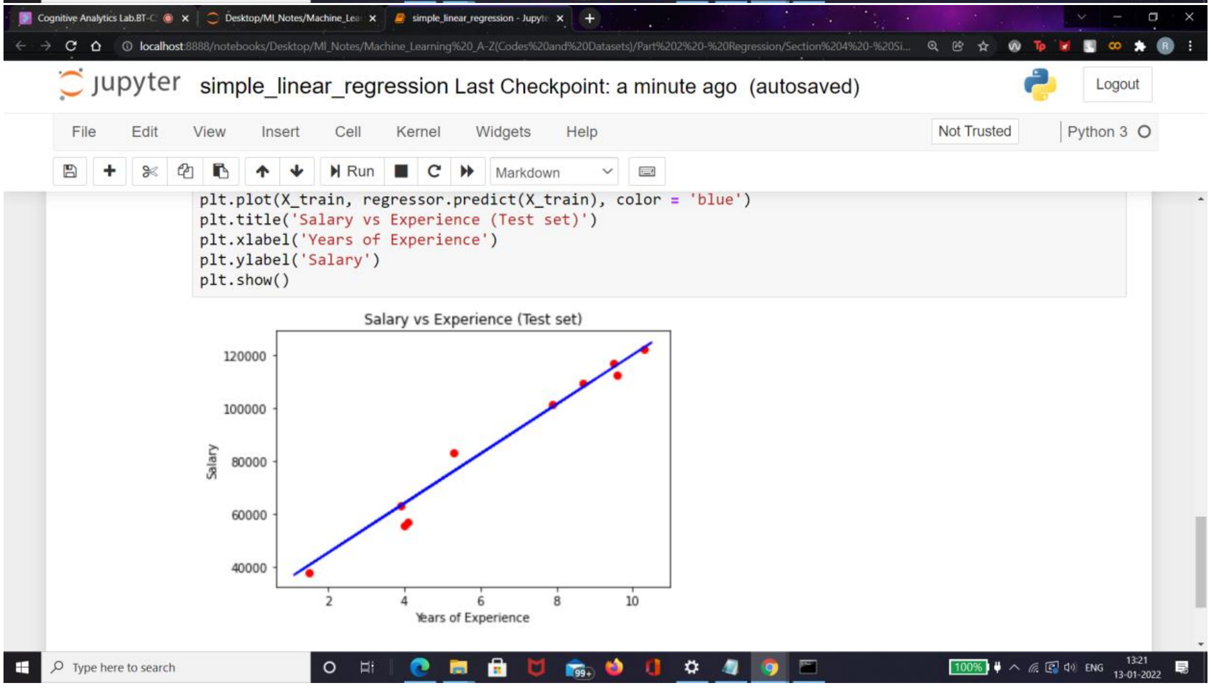
```
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```


Cognitive Analytics Lab.BT-C
Desktop/MI_Notes/Machine_Lea
simple_linear_regression - Jupyte
localhost:8888/notebooks/Desktop/MI_Notes/Machine_Learning%20_A-Z(Codes%20and%20Datasets)/Part%202%20-%20Regression/Section%204%20-%20Si...
jupyter
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Markdown
Salary vs Experience (Test set)
120000
100000
80000
60000
40000
Salary
Type here to search
100%
ENG
13:21
13-01-2022

simple_linear_regression Last Checkpoint: a minute ago (autosaved)

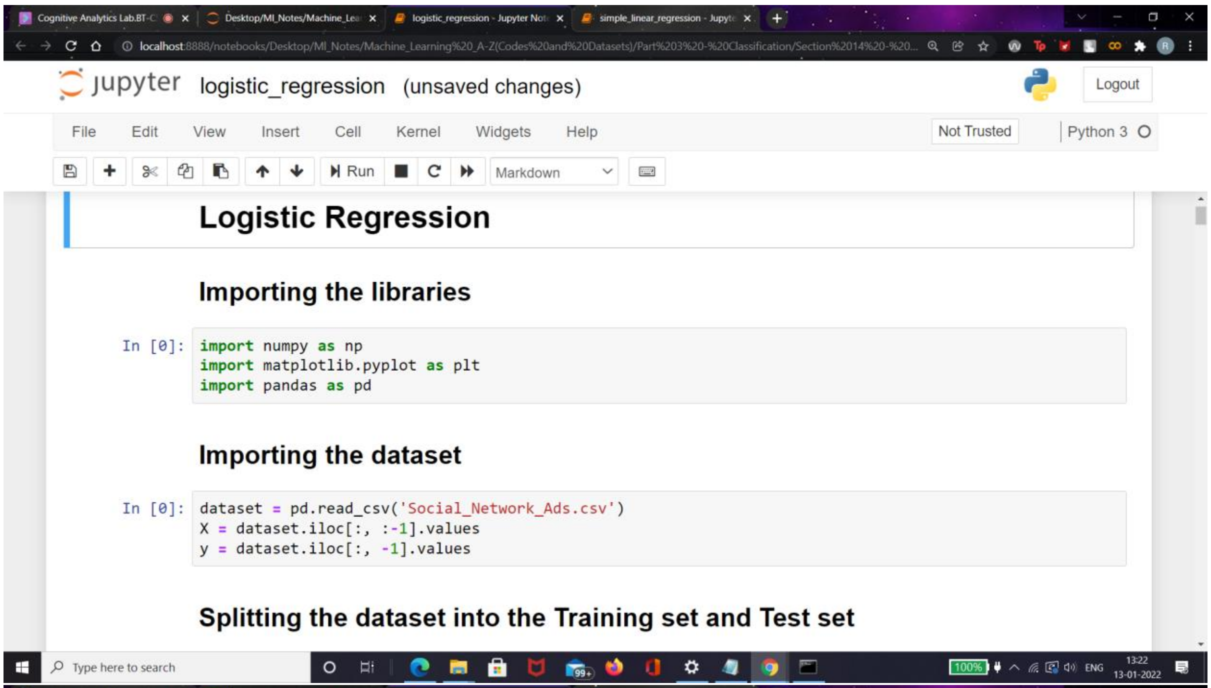
Visualising the Test set results

```
In [7]: 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()
```


Cognitive Analytics Lab.BT-C
Desktop/MI_Notes/Machine_Lea
simple_linear_regression - Jupyte
localhost:8888/notebooks/Desktop/MI_Notes/Machine_Learning%20_A-Z(Codes%20and%20Datasets)/Part%202%20-%20Regression/Section%204%20-%20Si...
jupyter
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Markdown
Salary vs Experience (Test set)
120000
100000
80000
60000
40000
Salary
2
4
6
8
10
Years of Experience
Type here to search
100%
ENG
13:21
13-01-2022

simple_linear_regression Last Checkpoint: a minute ago (autosaved)

```
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()
```


jupyter logistic_regression (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3
Run Markdown

Logistic Regression

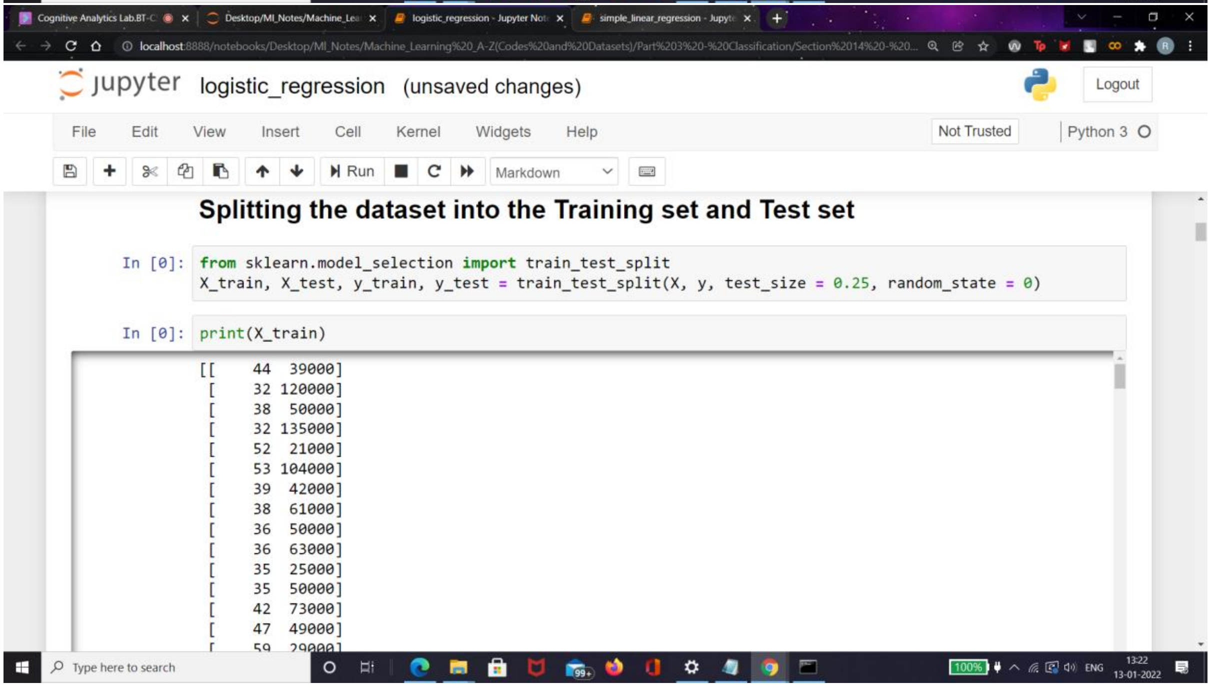
Importing the libraries

In [0]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

In [0]:
```python
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

Splitting the dataset into the Training set and Test set

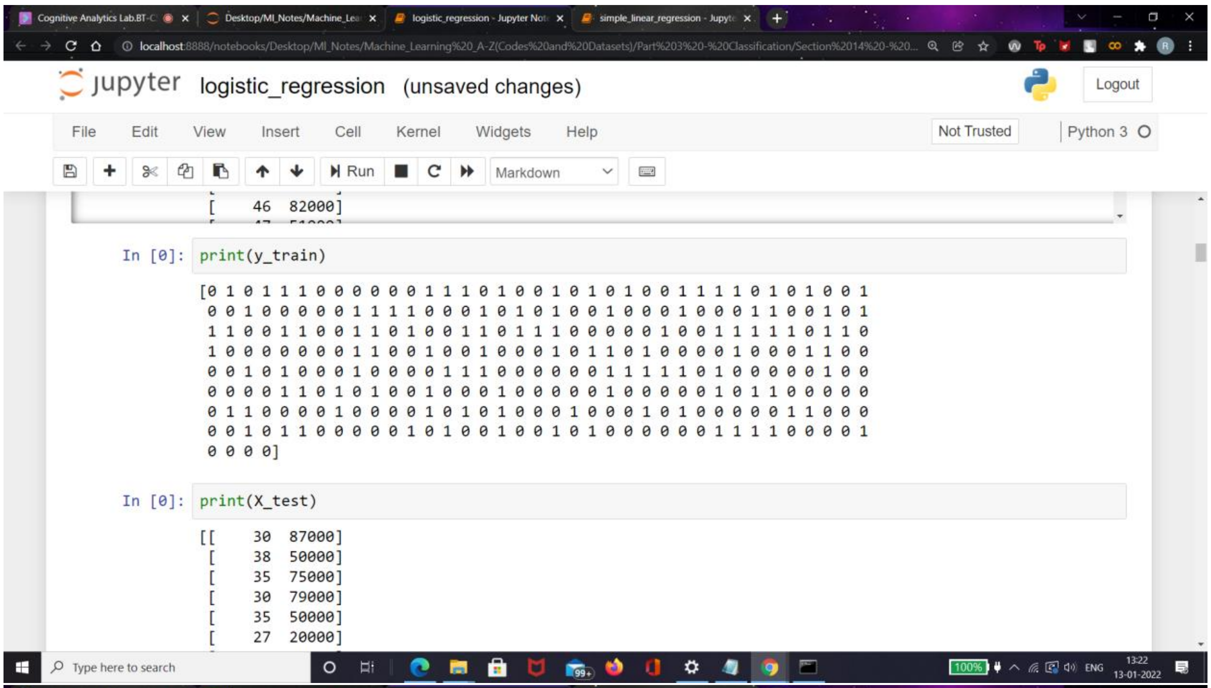

jupyter logistic_regression (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3
Run Markdown

Splitting the dataset into the Training set and Test set

In [0]:
```python
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)
```

In [0]:
```python
print(X_train)
```

```
[[    44  39000]
 [    32 120000]
 [    38  50000]
 [    32 135000]
 [    52  21000]
 [    53 104000]
 [    39  42000]
 [    38  61000]
 [    36  50000]
 [    36  63000]
 [    35  25000]
 [    35  50000]
 [    42  73000]
 [    47  49000]
 [    59  29000]
```


jupyter logistic_regression (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown

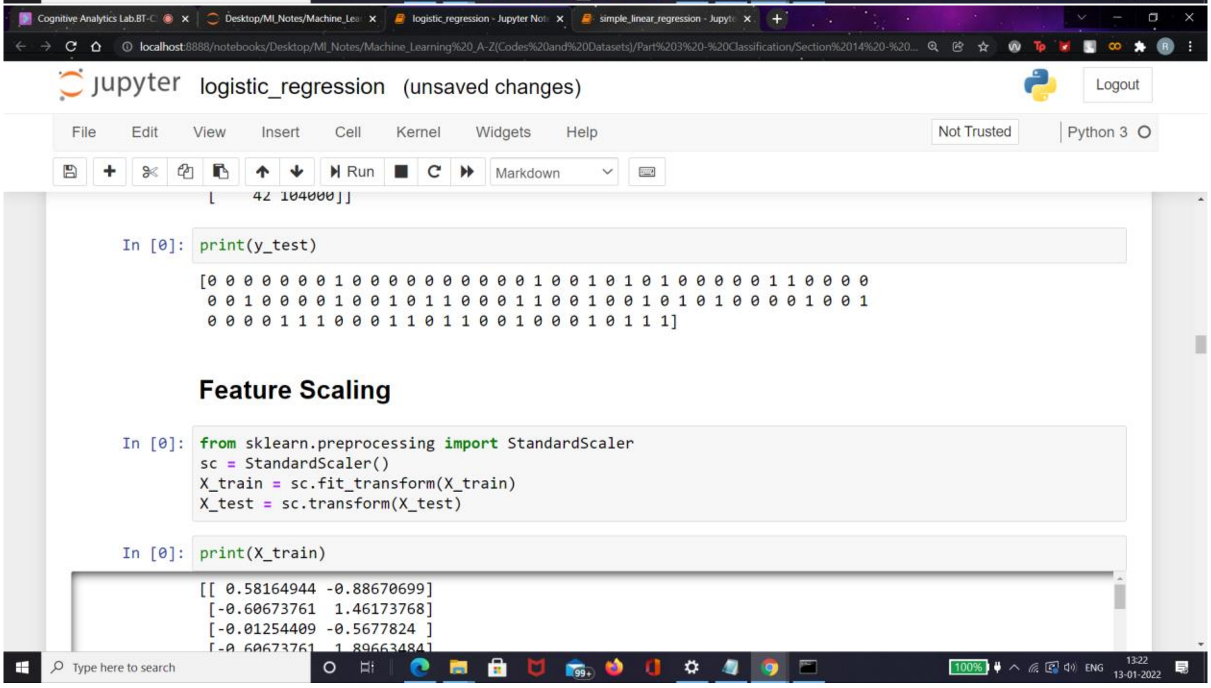
```
 [    46  82000]
```

In [0]:
```
print(y_train)
```

```
[0 1 0 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0 0 1
 0 0 1 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 1 1 0 0 1 0 1
 1 1 0 0 1 1 0 0 1 1 0 1 0 0 1 1 0 1 1 1 0 0 0 0 1 0 0 1 1 1 1 1 0 1 1 0
 1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0
 0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 1 0 0
 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 0 0 0 0
 0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0
 0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1
 0 0 0 0]
```

In [0]:
```
print(X_test)
```

```
[[    30  87000]
 [    38  50000]
 [    35  75000]
 [    30  79000]
 [    35  50000]
 [    27  20000]
```


jupyter logistic_regression (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown

```
 [    42 104000]]
```

In [0]:
```
print(y_test)
```

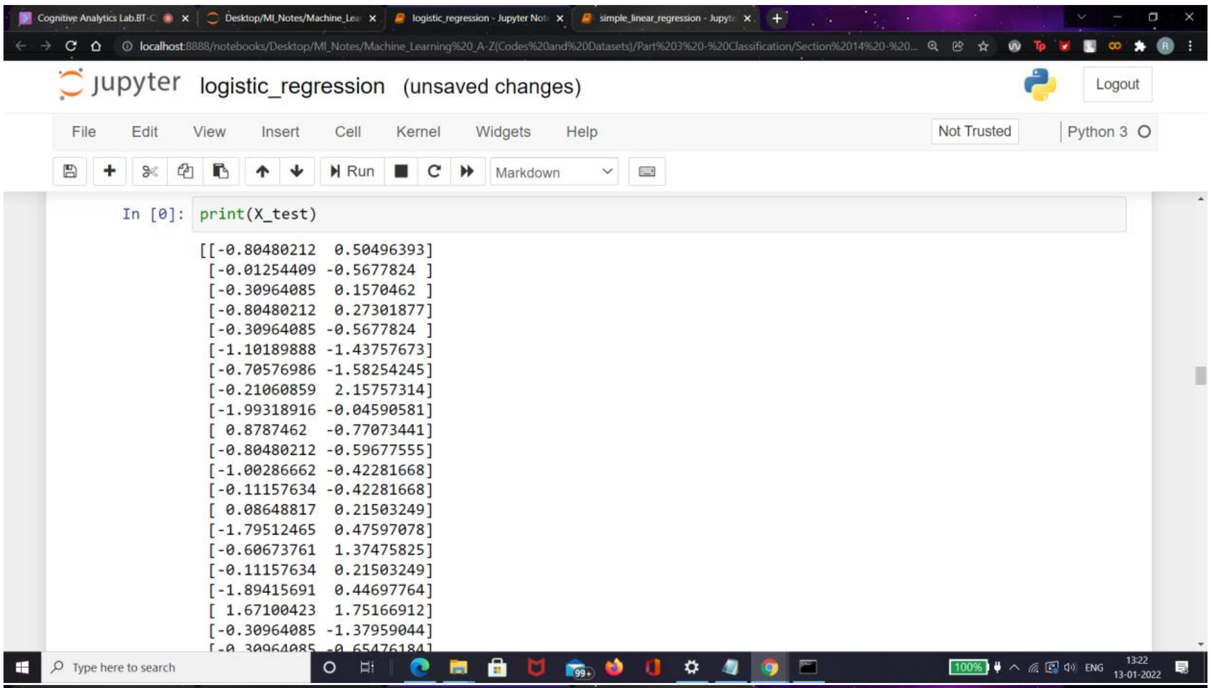
```
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 0
 0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1
 0 0 0 0 1 1 1 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 1 1 1]
```

Feature Scaling

In [0]:
```python
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

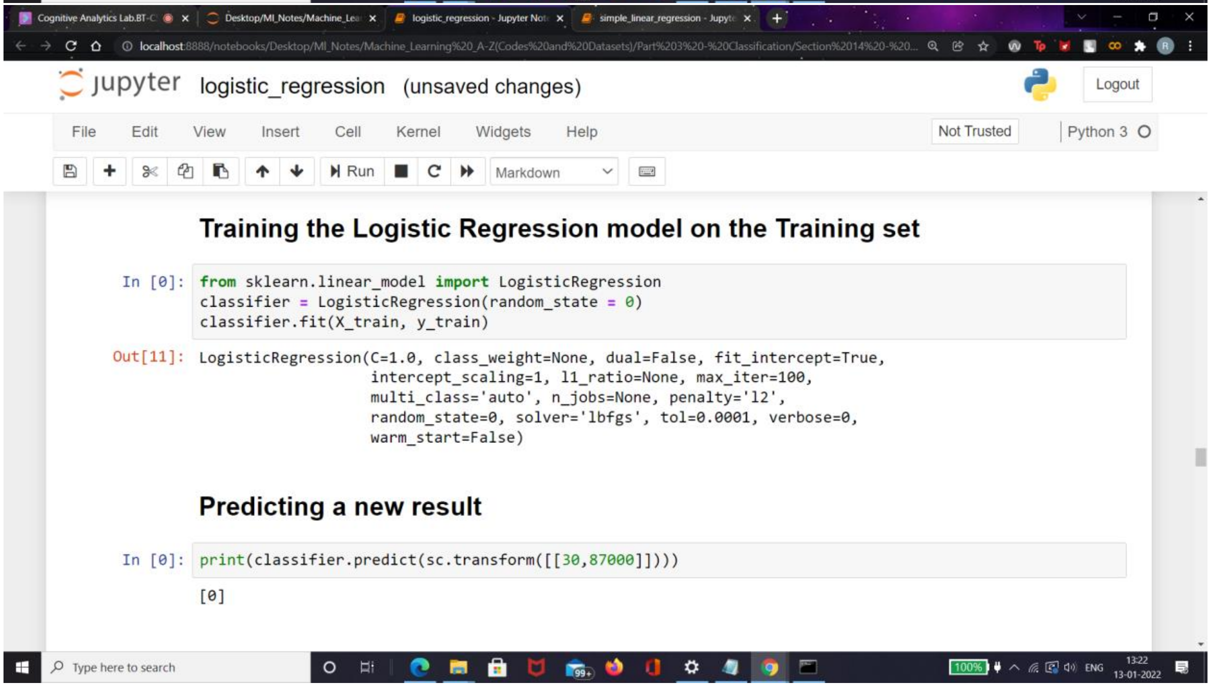
In [0]:
```
print(X_train)
```

```
[[ 0.58164944 -0.88670699]
 [-0.60673761  1.46173768]
 [-0.01254409 -0.5677824 ]
 [-0.60673761  1.89663484]
```


jupyter logistic_regression (unsaved changes)

In [0]:
```
print(X_test)
```

```
[[-0.80480212  0.50496393]
 [-0.01254409 -0.5677824 ]
 [-0.30964085  0.1570462 ]
 [-0.80480212  0.27301877]
 [-0.30964085 -0.5677824 ]
 [-1.10189888 -1.43757673]
 [-0.70576986 -1.58254245]
 [-0.21060859  2.15757314]
 [-1.99318916 -0.04590581]
 [ 0.8787462  -0.77073441]
 [-0.80480212 -0.59677555]
 [-1.00286662 -0.42281668]
 [-0.11157634 -0.42281668]
 [ 0.08648817  0.21503249]
 [-1.79512465  0.47597078]
 [-0.60673761  1.37475825]
 [-0.11157634  0.21503249]
 [-1.89415691  0.44697764]
 [ 1.67100423  1.75166912]
 [-0.30964085 -1.37959044]
 [-0.30964085 -0.65476184]
```


jupyter logistic_regression (unsaved changes)

Training the Logistic Regression model on the Training set

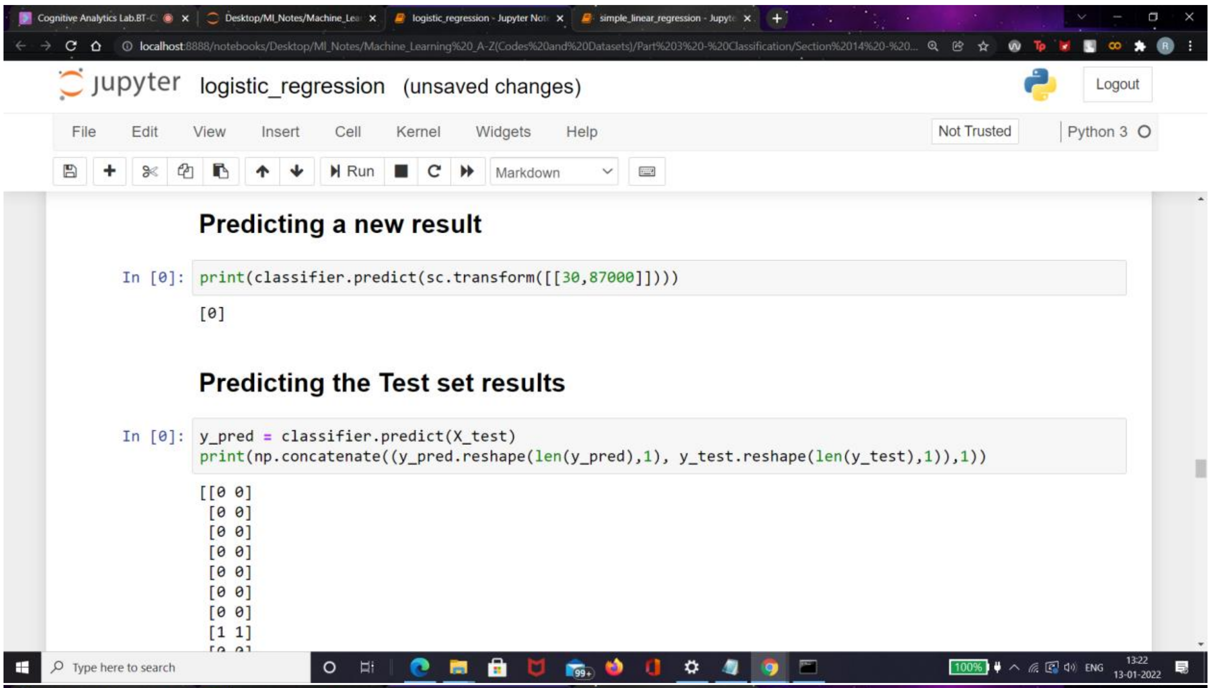
In [0]:
```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

Out[11]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Predicting a new result

In [0]:
```
print(classifier.predict(sc.transform([[30,87000]])))
```

```
[0]
```

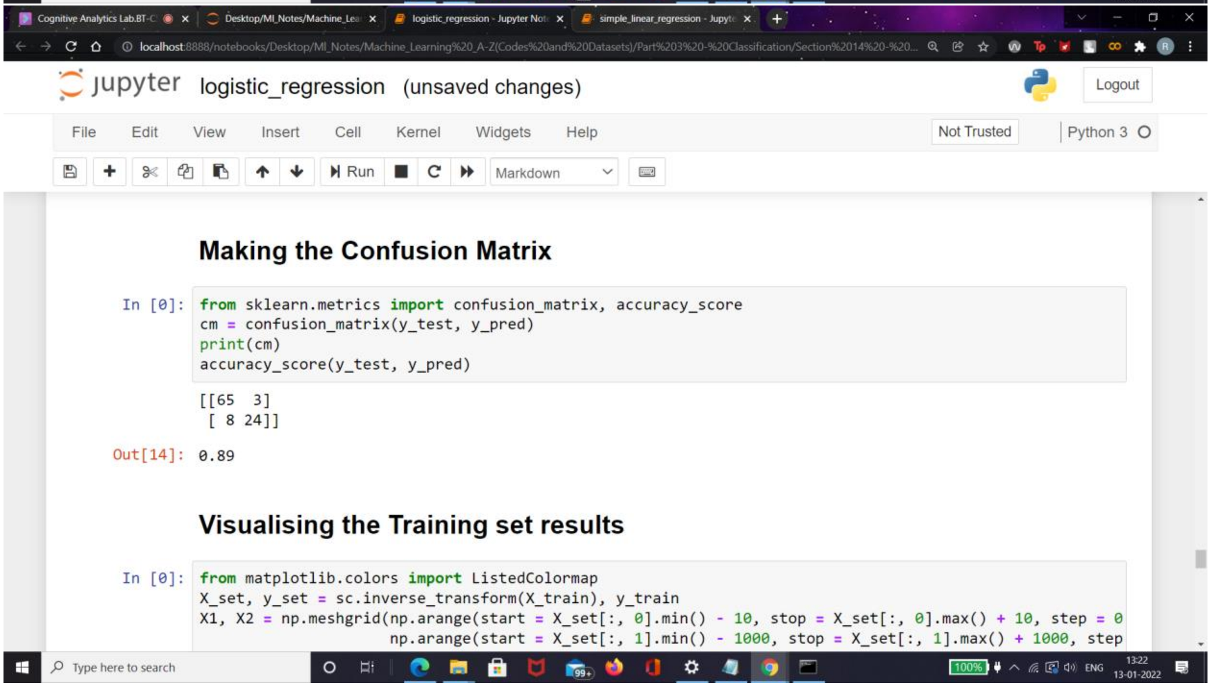

jupyter
logistic_regression (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown

Predicting a new result

```
In [0]: print(classifier.predict(sc.transform([[30,87000]])))

        [0]
```

Predicting the Test set results

```
In [0]: y_pred = classifier.predict(X_test)
        print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))

        [[0 0]
         [0 0]
         [0 0]
         [0 0]
         [0 0]
         [0 0]
         [0 0]
         [1 1]
```

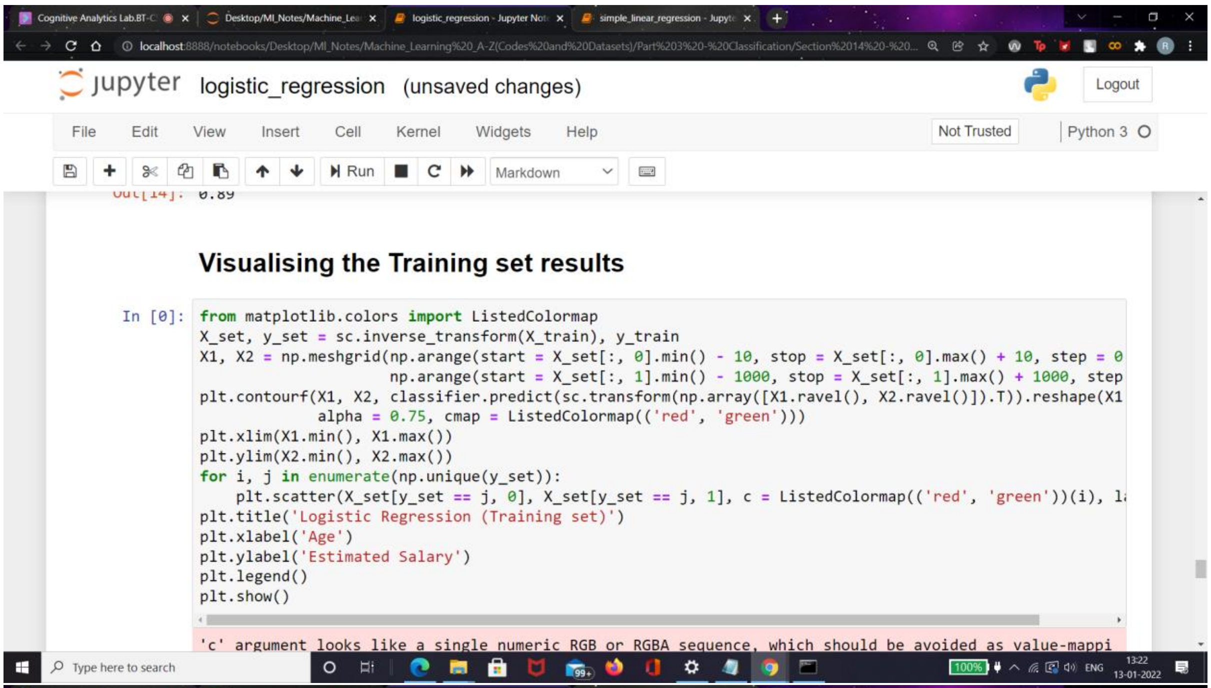

jupyter
logistic_regression (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown

Making the Confusion Matrix

```
In [0]: from sklearn.metrics import confusion_matrix, accuracy_score
        cm = confusion_matrix(y_test, y_pred)
        print(cm)
        accuracy_score(y_test, y_pred)

        [[65  3]
         [ 8 24]]

Out[14]: 0.89
```

Visualising the Training set results

```
In [0]: from matplotlib.colors import ListedColormap
        X_set, y_set = sc.inverse_transform(X_train), y_train
        X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, step = 0
                             np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step
```

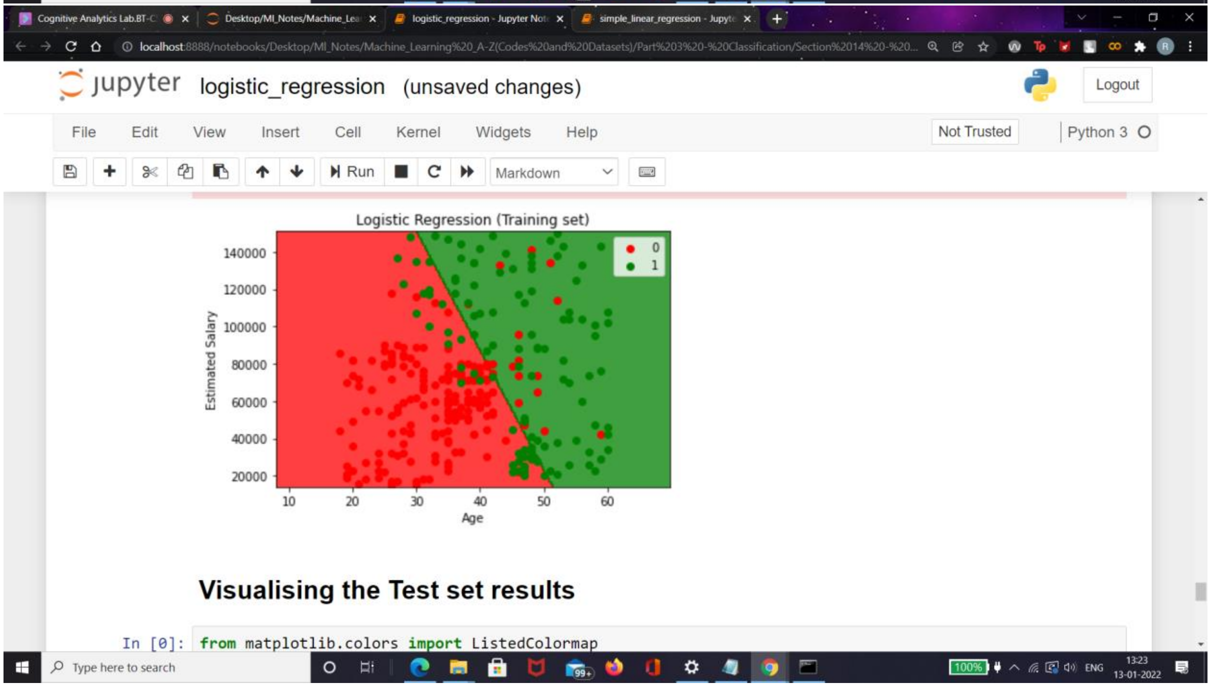

Cognitive Analytics Lab.BT-C
Desktop/Ml_Notes/Machine_Lea
logistic_regression - Jupyter Note
simple_linear_regression - Jupyte
localhost:8888/notebooks/Desktop/Ml_Notes/Machine_Learning%20_A-Z(Codes%20and%20Datasets)/Part%203%20-%20Classification/Section%2014%20-%20...
jupyter logistic_regression (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
Type here to search
13:22
13-01-2022

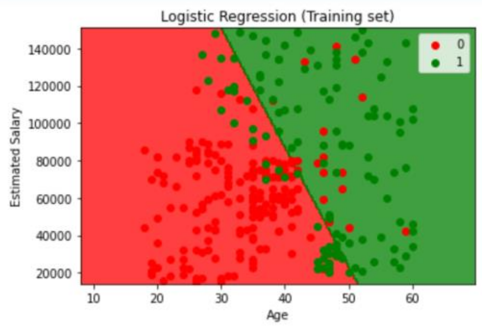
```
Out[14]: 0.89
```

Visualising the Training set results

```
In [0]: from matplotlib.colors import ListedColormap
X_set, y_set = sc.inverse_transform(X_train), y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, step = 0
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(i), la
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

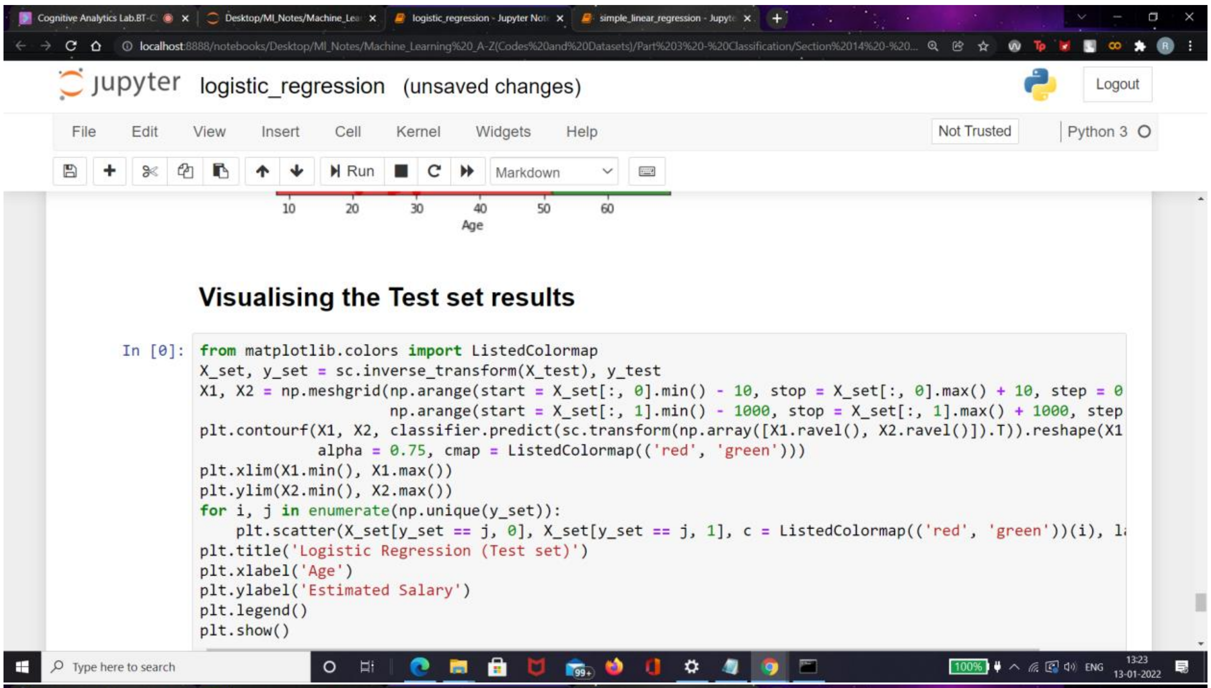
```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mappi
```


Cognitive Analytics Lab.BT-C
Desktop/Ml_Notes/Machine_Lea
logistic_regression - Jupyter Note
simple_linear_regression - Jupyte
localhost:8888/notebooks/Desktop/Ml_Notes/Machine_Learning%20_A-Z(Codes%20and%20Datasets)/Part%203%20-%20Classification/Section%2014%20-%20...
jupyter logistic_regression (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
Type here to search
13:23
13-01-2022



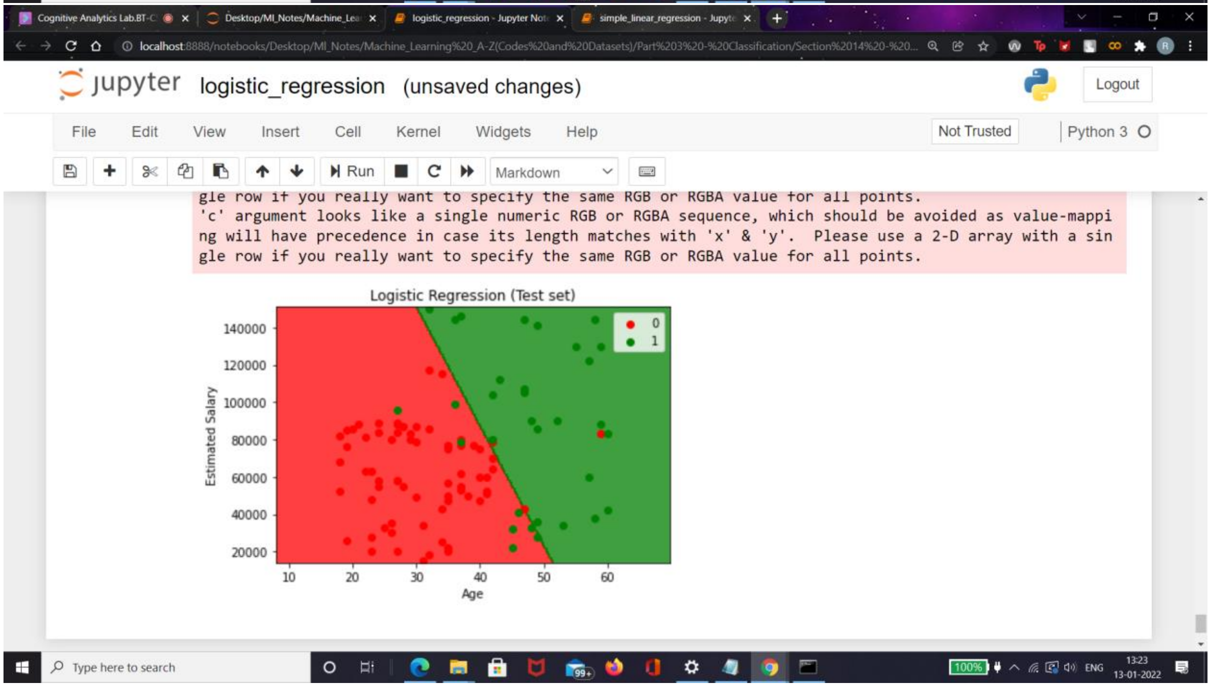
Visualising the Test set results

```
In [0]: from matplotlib.colors import ListedColormap
```



Visualising the Test set results

```
In [0]: from matplotlib.colors import ListedColormap
        X_set, y_set = sc.inverse_transform(X_test), y_test
        X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 10, stop = X_set[:, 0].max() + 10, step = 0
                             np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step
        plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1
                     alpha = 0.75, cmap = ListedColormap(('red', 'green')))
        plt.xlim(X1.min(), X1.max())
        plt.ylim(X2.min(), X2.max())
        for i, j in enumerate(np.unique(y_set)):
            plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('red', 'green'))(i), la
        plt.title('Logistic Regression (Test set)')
        plt.xlabel('Age')
        plt.ylabel('Estimated Salary')
        plt.legend()
        plt.show()
```



```
gle row if you really want to specify the same RGB or RGBA value for all points.
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mappi
ng will have precedence in case its length matches with 'x' & 'y'.  Please use a 2-D array with a sin
gle row if you really want to specify the same RGB or RGBA value for all points.
```