Using Logistic Regression model to predict if a person is going to buy a new car or not based on dataset

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from matplotlib.colors import ListedColormap
from sklearn import metrics
```

```
from google.colab import files
data_to_load = files.upload()
```

Choose Files Social_Network_Ads.csv

- **Social_Network_Ads.csv**(n/a) - 10930 bytes, last modified: 12/26/2019 - 100% done

```
Saving Social_Network_Ads.csv to Social_Network_Ads (5).csv
```

The dataset contains the userId, gender, age, estimatedsalary and the purchased history. The matrix of features taken into account are age and estimated salary which are going to predict if the user is going to buy new car or not(1=Yes, 0=No).

```
import io
dataset = pd.read_csv(io.BytesIO(data_to_load['Social_Network_Ads.csv']))
dataset.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19.0	19000.0	0
1	15810944	Male	35.0	20000.0	0
2	15668575	Female	26.0	43000.0	0
3	15603246	Female	27.0	57000.0	0
4	15804002	Male	19.0	76000.0	0

```
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
log_reg = LogisticRegression(random_state = 0)
log_reg.fit(X_train, y_train)
```

```
     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)
```

```
y_pred = log_reg.predict(X_test)
```

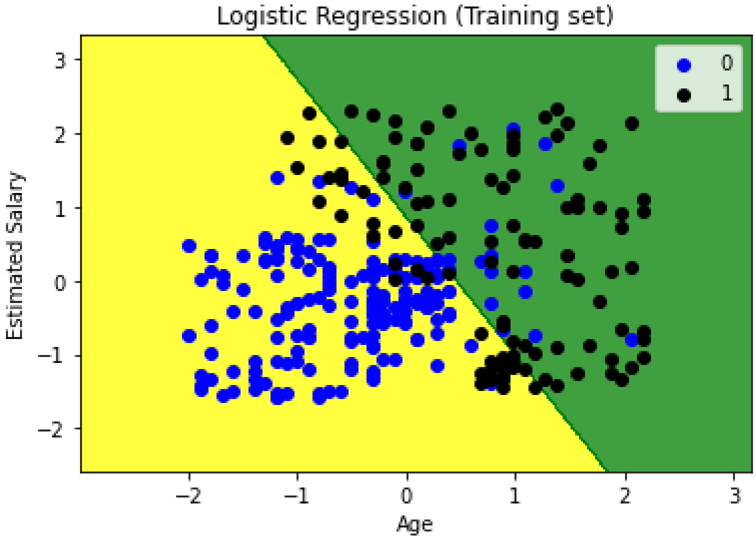
The confusion matrix below shows that our model predicts 89 correct and 11 wrong decisions which shows 89% accuracy.

```
cm = confusion_matrix(y_test, y_pred)
cm
```

```
     array([[65,  3],
            [ 8, 24]])
```

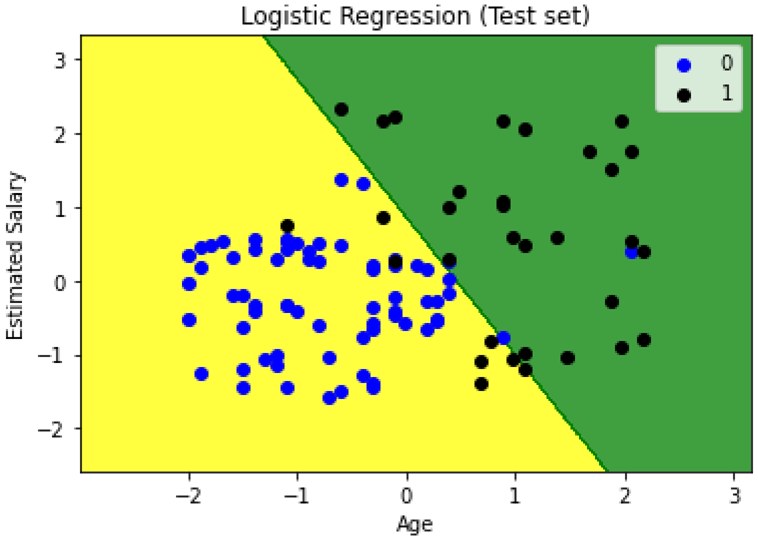
```
from matplotlib.colors import ListedColormap
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0
plt.contourf(X1, X2, log_reg.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('yellow', '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(('blue', 'black'))(i), label = j)
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 valu
     *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as valu
```



```
from matplotlib.colors import ListedColormap
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step =
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0
plt.contourf(X1, X2, log_reg.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('yellow', '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(('blue', 'black'))(i), label = j)
plt.title('Logistic Regression (Test 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
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value
```



The data visualization of the training set and test set is given above. As logistic regression is linear model the data is being separated linearly. The black dots shows the people buying the car whereas blue dots shows the people who don't buy the car.

```
model = LogisticRegression()
model.fit(X_train,y_train)
```

```
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=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
lr_prediction = model.predict(X_test)
print('Logistic Regression accuracy = ', metrics.accuracy_score(lr_prediction,y_test))
```

```
Logistic Regression accuracy =  0.89
```

✓ 0s completed at 3:16 PM ● ✕