ML EXPERIMENT 3

LOGISTIC REGRESSION

Importing the libraries

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

Importing the dataset

```
[ ] 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

```
[ ] 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)
```

Feature Scaling

Feature Scaling

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

Training the Logistic Regression model on the Training set

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

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

Predicting a new result

```
[ ] classifier.predict(sc.transform([[30, 87000]]))

array([0])
```

Making the Confusion Matrix

```
[ ] from sklearn.metrics import confusion_matrix, accuracy_score
    cm = confusion_matrix(y_test,y_pred)
    print(cm) # Gives us the confusion matrix
    accuracy_score(y_test, y_pred) # Rate of correct prediction

[[65  3]
 [ 8 24]]
0.89
```

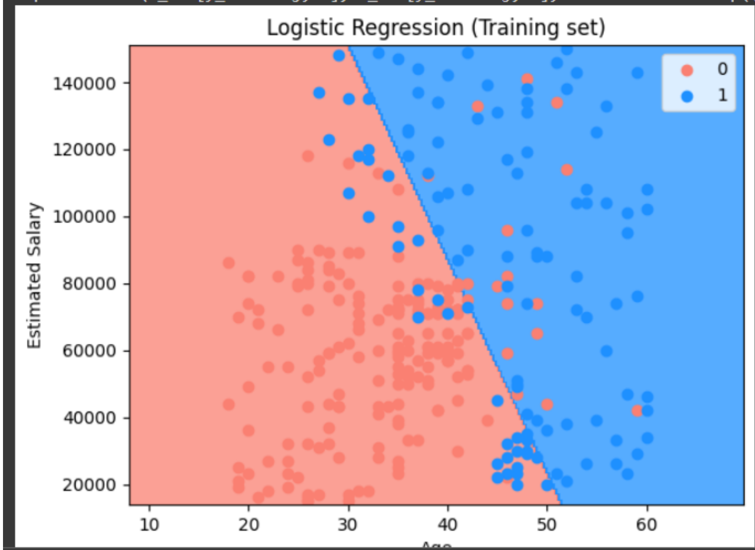
+ Code + Text

Visualising the Training set results

```
[ ] # By this entire code we can see that the curve here is a straight line since logistic regression is a linear classifier
    # We can also see that the demarkation has been made by the color scope showing the results
    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 = 0.25),
                         np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step = 0.25))
    plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1.shape),
                 alpha = 0.75, cmap = ListedColormap(('salmon', 'dodgerblue')))
    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(('salmon', 'dodgerblue'))(i), label = j)
plt.title('Logistic Regression (Training set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
<ipython-input-9-498cf0fd9ad8>:10: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should b
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('salmon', 'dodgerblue'))(i), label = j)
```



Connected to Python 3 Google Compute Engine backend (GPU)

```
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 = 0.25),
                     np.arange(start = X_set[:, 1].min() - 1000, stop = X_set[:, 1].max() + 1000, step = 0.25))
plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), X2.ravel()]).T)).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('salmon', 'dodgerblue')))
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(('salmon', 'dodgerblue'))(i), label = j)
plt.title('Logistic Regression (Test set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
<ipython-input-10-96b67a383660>:10: UserWarning: *c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapp
  plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1], c = ListedColormap(('salmon', 'dodgerblue'))(i), label = j)
```