8. Logistic Regression

- Logistic Regression is an example of a classification algorithm which is used to find a relationship between features and probability of a particular outcome.
- Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y).
- A key difference from linear regression is that the output value being modeled is a binary values (0 or 1) rather than a numeric value.
- logistic regression equation:

y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x))

Problem Statement - To predict whether a person will buy a car (1) or (0)

Dataset description

We have a Data set having 5 columns namely: User ID, Gender, Age, EstimatedSalary and Purchased. Now we have to build a model that can predict whether on the given parameter a person will buy a car or not.

Importing the libraries

In [ ]:

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

Importing the Data set

In [ ]:

```
dataset = pd.read_csv('Social_Network_Ads.csv')
```

In [ ]:

```
dataset.shape
```

Out[ ]:

```
(400, 5)
```

In [ ]:

```
dataset.head(5)
```

Out[ ]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

Train- Test split

Splitting our Data set in Dependent and Independent variables.

In our Data set we'll consider Age and EstimatedSalary as Independent variable and Purchased as Dependent Variable.

In [ ]:

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

Here X is Independent variable and y is Dependent variable.Logistic model and Test data will be used to validate our model. We'll use Sklearn to split our data. We'll import train_test_split from sklearn.model_selection

In [ ]:

```
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_stat
e = 0)
```

Feature Scaling

we'll do feature scaling to scale our data between 0 and 1 to get better accuracy. Here Scaling is important because there is a huge difference between Age and EstimatedSalay.

- Import StandardScaler from sklearn.preprocessing
- Then make an instance sc_X of the object StandardScaler
- Then fit and transform X_train and transform X_test

In [ ]:

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

Fitting Logistic Regression to the Training Set

In [ ]:

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

Out[ ]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
                   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 the Test set results

In [ ]:

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

In [ ]:

```
y_pred
```

Out[ ]:

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1])
```

Making the confusion matrix

In [ ]:

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

In [ ]:

```
cm
```

Out[ ]:

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

In [ ]:

```
import sklearn
sklearn.metrics.accuracy_score(y_test, y_pred)*100
```

Out[ ]:

```
89.0
```

In [1]:

```
print("VINITA VERMA ")
print("MCA-5B")
print("06917704418")
```

```
VINITA VERMA
MCA-5B
06917704418
```

In [ ]:

```

```