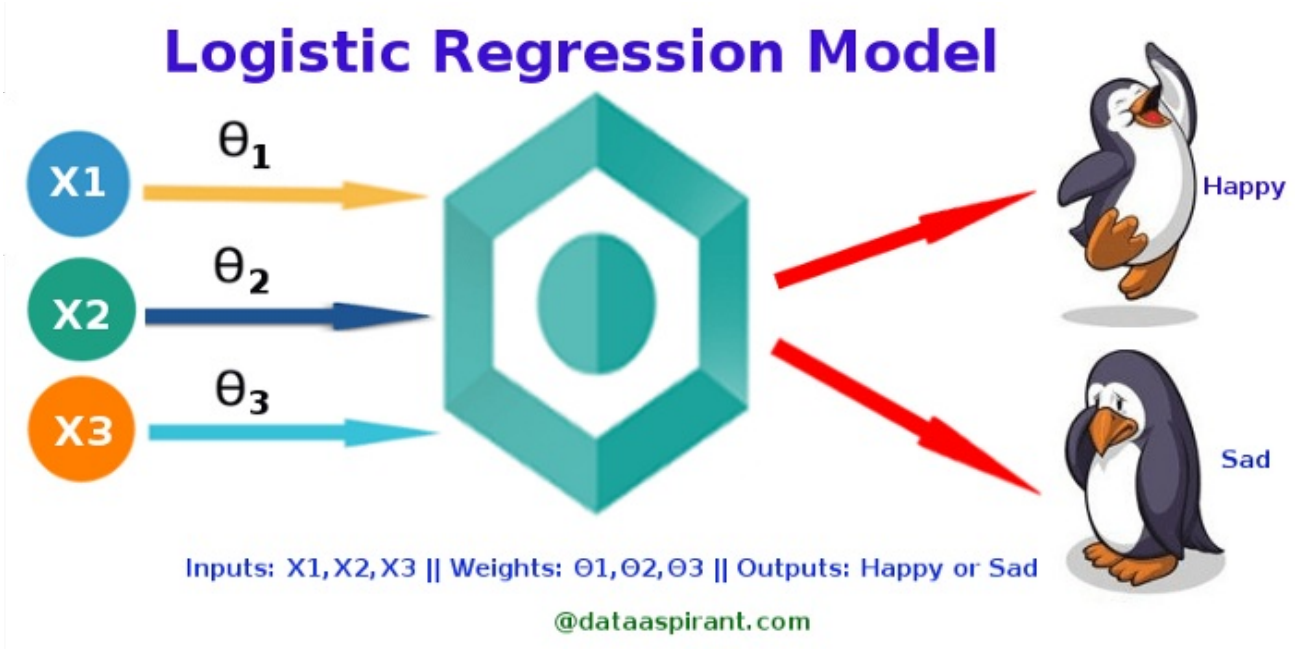
Logistic Regression

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).



Importing Common Libraries

In [1]:

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

In [2]:

```
train=pd.read_csv("Downloads/train.csv")
test=pd.read_csv("Downloads/test.csv")
```

In [3]:

```
train.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William								

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Percentage Of Null Values

In [4]:

```
train.isnull().sum()*100/len(train)
```

Out[4]:

```
PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            19.865320
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.000000
Cabin          77.104377
Embarked        0.224467
dtype: float64
```

In [5]:

```
#cabin contain more null values than data so, can drop cabin column for better modeling
train= train.drop(["Cabin"], axis=1)
```

In [6]:

```
#filling age null values with mean
train['Age'].fillna(train['Age'].mean(), inplace=True)
#filling Embarked null values with mode
train['Embarked'].fillna(train['Embarked'].mode(), inplace=True)
```

In [7]:

```
train.dtypes
```

Out[7]:

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Embarked        object
dtype: object
```

In [8]:

```
#drop PassengerId, Name,Ticket,Fare Since,have no use in predicting target feature(Survived)
train = train.drop(["PassengerId", "Name", "Ticket", "Fare"], axis = 1)
```

Label Encoding

In [9]:

```
# Categorical boolean mask
categorical_feature_mask = train.dtypes==object
```

```
# filter categorical columns using mask and turn it into a list
categorical_cols = train.columns[categorical_feature_mask].tolist()
print(categorical_cols)
```

```
['Sex', 'Embarked']
```

In [10]:

```
# import labelencoder
from sklearn.preprocessing import LabelEncoder
# instantiate labelencoder object
le = LabelEncoder()
train[categorical_cols[0]] = le.fit_transform(train[categorical_cols[0]].astype('str'))
train[categorical_cols[1]] = le.fit_transform(train[categorical_cols[1]].astype('str'))
```

Splitting dataset

In [11]:

```
X = train.drop(['Survived'], axis=1)
y = train['Survived']
```

In [12]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

Logistic Regression Model

In [13]:

```
from sklearn.linear_model import LogisticRegression

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

Out[13]:

```
LogisticRegression()
```

In [14]:

```
y_pred = logReg.predict(X_test)

prediction= pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
prediction.head()
```

Out[14]:

	Actual	Predicted
646	0	0
334	1	1
596	1	1
383	1	1
375	1	1

Model Accuracy

In [15]:

```
print("Training accuracy: ",logReg.score(X_train,y_train))
```

```
print("Test accuracy: ",logReg.score(X_test,y_test))
```

```
Training accuracy:  0.7963483146067416
Test accuracy:  0.8044692737430168
```

Prediction for Test dataset

In [16]:

```
test.head()
```

Out[16]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

Percentage of null vaules in test

In [17]:

```
test.isnull().sum()*100/len(test)
```

Out[17]:

```
PassengerId     0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            20.574163
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.239234
Cabin          78.229665
Embarked        0.000000
dtype: float64
```

In [18]:

```
#cabin contain more null values than data so, can drop cabin column for better modeling
test= test.drop(["Cabin"], axis=1)
```

In [19]:

```
#filling age null values with mean
test['Age'].fillna(test['Age'].mean(), inplace=True)
#filling Fare null values with mode
test['Fare'].fillna(test['Fare'].mode(), inplace=True)
```

In [20]:

```
test.dtypes
```

Out[20]:

```
PassengerId      int64
Pclass           int64
Name            object
Sex             object
Age            float64
```

```
SibSp             int64
Parch             int64
Ticket           object
Fare            float64
Embarked         object
dtype: object
```

In [21]:

```
#drop PassengerId, Name,Ticket,Fare Since,have no use in predicting target feature(Survived)
Id = test["PassengerId"]
test = test.drop(["PassengerId", "Name", "Ticket", "Fare"], axis = 1)
```

Label Encoding

In [22]:

```
# Categorical boolean mask
categorical_feature_mask_test = test.dtypes==object
# filter categorical columns using mask and turn it into a list
categorical_cols_test = test.columns[categorical_feature_mask_test].tolist()
print(categorical_cols_test)
```

```
['Sex', 'Embarked']
```

In [23]:

```
test[categorical_cols_test[0]] = le.fit_transform(test[categorical_cols_test[0]].astype('str'))
test[categorical_cols_test[1]] = le.fit_transform(test[categorical_cols_test[1]].astype('str'))
```

In [24]:

```
test.head()
```

Out[24]:

	Pclass	Sex	Age	SibSp	Parch	Embarked
0	3	1	34.5	0	0	1
1	3	0	47.0	1	0	2
2	2	1	62.0	0	0	1
3	3	1	27.0	0	0	2
4	3	0	22.0	1	1	2

Prediction for test dataset

In [25]:

```
y_test_pred = logReg.predict(test)
```

In [26]:

```
final_result = pd.DataFrame({
        "PassengerId": Id,
        "Survived": y_test_pred
    })

print(final_result.head())
```

```
   PassengerId  Survived
0          892         0
1          893         0
```

```
2           894         0
3           895         0
4           896         1
```

In [ ]:

```
```