WEEK-4

Aim: Write a program to implement logistic regression model (eg: Titanic shipwreck https://www.kaggle.com/c/titanic)

Description:

What is logistic regression?

Logistic regression is an example of supervised learning. It is used to calculate or predict the probability of a binary (yes/no) event occurring. An example of logistic regression could be applying machine learning to determine if a person is likely to be infected with COVID-19 or not. Since we have two possible outcomes to this question - yes they are infected, or no they are not infected - this is called binary classification.

The three types of logistic regression

1. **Binary logistic regression** - When we have two possible outcomes, like our original example of whether a person is likely to be infected with COVID-19 or not.
2. **Multinomial logistic regression** - When we have multiple outcomes, say if we build out our original example to predict whether someone may have the flu, an allergy, a cold, or COVID-19.
3. **Ordinal logistic regression** - When the outcome is ordered, like if we build out our original example to also help determine the severity of a COVID-19 infection, sorting it into mild, moderate, and severe cases.

Training data assumptions for logistic regression

Training data that satisfies the below assumptions is usually a good fit for logistic regression.

- The predicted outcome is strictly binary or dichotomous. (This applies to binary logistic regression).
- The factors, or the independent variables, that influence the outcome are independent of each other. In other words there is little or no multicollinearity among the independent variables.
- The independent variables can be linearly related to the log odds.
- Fairly large sample sizes.

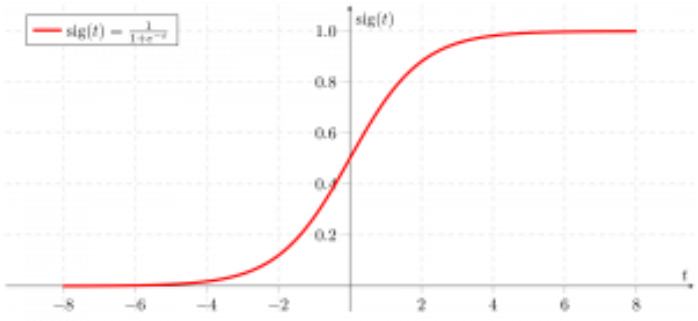
Logistic Function

$$h\Theta(x) = 1/1 + e - (\beta o + \beta 1X)$$

'hΘ(x)' *is output of logistic function , where 0 ≤ hΘ(x) ≥ 1*

*'**β1**' is the slope*
*'**βo**' is the y-intercept*
*'**X**' is the independent variable*

*(βo + β1*x) - derived from equation of a line Y(predicted) = (βo + β1*x) + Error value*



Code:

```
#importing libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns


#reading titanic dataset

df = pd.read_csv("/content/drive/MyDrive/isl/titanic/titanic.csv")

df.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 Henry	male	35.0	0	0	373450	8.0500	NaN	S

#descriptive statistics of numeric variables

df.describe()

Out[5]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

df.describe(include = ['O'])

Out[6]:

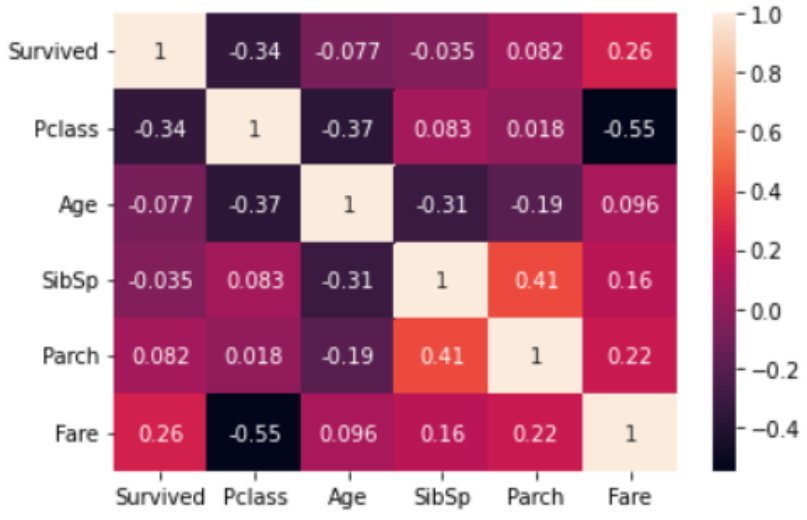
	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Braund, Mr. Owen Harris	male	347082	B96 B98	S
freq	1	577	7	4	644

df_corr = df.corr()

Correlation Heatmap

sns.heatmap(df_corr, annot=True)

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7f0295a42f50>



```
In [10]: df.isna().sum()

Out[10]: Survived      0
         Pclass        0
         Sex           0
         Age         177
         SibSp         0
         Parch         0
         Fare          0
         Embarked      2
         dtype: int64
```

Filling null values with mean

df.Age.fillna(df['Age'].mean(), inplace=True)

Filling null values with mode

df.Embarked.replace('NaN', np.nan, inplace=True)

df['Embarked'].fillna(list(df['Embarked'].mode())[0], inplace=True)

```
In [14]: df.isna().sum()

Out[14]: Survived    0
         Pclass      0
         Sex         0
         Age         0
         SibSp       0
         Parch       0
         Fare        0
         Embarked    0
         dtype: int64
```

Categorical to numeric

dummies = pd.get_dummies(df['Embarked'])

df = pd.concat([df, dummies], axis=1)

df.head()

Out[17]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	C	Q	S
0	0	3	male	22.0	1	0	7.2500	S	0	0	1
1	1	1	female	38.0	1	0	71.2833	C	1	0	0
2	1	3	female	26.0	0	0	7.9250	S	0	0	1
3	1	1	female	35.0	1	0	53.1000	S	0	0	1
4	0	3	male	35.0	0	0	8.0500	S	0	0	1

```
dummies = pd.get_dummies(df['Sex'])

df = pd.concat([df, dummies], axis=1)

df.head()
```

Out[19]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	C	Q	S	female	male
0	0	3	male	22.0	1	0	7.2500	S	0	0	1	0	1
1	1	1	female	38.0	1	0	71.2833	C	1	0	0	1	0
2	1	3	female	26.0	0	0	7.9250	S	0	0	1	1	0
3	1	1	female	35.0	1	0	53.1000	S	0	0	1	1	0
4	0	3	male	35.0	0	0	8.0500	S	0	0	1	0	1

```
df.drop(['Sex', 'Embarked'], axis=1, inplace=True)

df.head()
```

Out[20]:

	Survived	Pclass	Age	SibSp	Parch	Fare	C	Q	S	female	male
0	0	3	22.0	1	0	7.2500	0	0	1	0	1
1	1	1	38.0	1	0	71.2833	1	0	0	1	0
2	1	3	26.0	0	0	7.9250	0	0	1	1	0
3	1	1	35.0	1	0	53.1000	0	0	1	1	0
4	0	3	35.0	0	0	8.0500	0	0	1	0	1

```
# Standardising Data

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
```

```
scaler_series = scaler.fit_transform(df[['Age', 'Fare']])

scaler_df = pd.DataFrame(scaler_series, columns=['Age', 'Fare'])

df.drop(['Age', 'Fare'], axis=1, inplace=True)

df = pd.concat([df, scaler_df], axis=1)

df.head()
```

Out[21]:

	Survived	Pclass	SibSp	Parch	C	Q	S	female	male	Age	Fare
0	0	3	1	0	0	0	1	0	1	-0.592481	-0.502445
1	1	1	1	0	1	0	0	1	0	0.638789	0.786845
2	1	3	0	0	0	0	1	1	0	-0.284663	-0.488854
3	1	1	1	0	0	0	1	1	0	0.407926	0.420730
4	0	3	0	0	0	0	1	0	1	0.407926	-0.486337

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

y = df['Survived']


# Model training

from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(random_state=0)

clf.fit(X, y)
```

```
Out[23]: LogisticRegression(random_state=0)
```

```
# Model Score

clf.score(X, y)
```

```
In [24]: # Model Score
         clf.score(X, y)

Out[24]: 0.8013468013468014
```