Assignment No: 5

Title:

1. Implement logistic regression using Python/R to perform classification on Social_Network_Ads.csv dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

Objective: Students should be able to data analysis using logisticregression using Python for any open-source dataset

Prerequisite:

1. Basic of Python Programming

2.Concept of Regression.

Contents for Theory:

1. Logistic Regression
2. Differentiate between Linear and Logistic Regression
3. Sigmoid Function
4. Types of Logistic Regression
5. Confusion Matrix Evaluation Metrics

1. **Logistic Regression:** Classification techniques are an essential part of machine learning and data mining applications. Approximately 70% of problems in Data Science are classification problems. There are lots of classification problems that are available, but logistic regression is common and is a useful regression method for solving the binary classification problem. Another category of classification is Multinomial classification, which handles the issues where multiple classes are present in the target variable. For example, the IRIS dataset is a very famous example of multi-class classification. Other examples are classifying article/blog/document categories.

Logistic Regression can be used for various classification problems such as spam detection. Diabetes prediction, if a given customer will purchase a particular product or will they churn another competitor, whether the user will click on a given advertisement link or not, and many more examples are in the bucket.

Logistic Regression is one of the most simple and commonly used Machine Learning algorithms for two-class classification. It is easy to implement and can be used as the baseline for any binary classification problem. Its basic fundamental concepts are also constructive in deep learning. Logistic regression describes and estimates the relationship between one dependent binary variable and independent variables.

Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for cancer detection problems. It computes the probability of an event occurring.

It is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilising a logit function.

Linear Regression Equation:

$$y = \beta 0 + \beta 1X1 + \beta 2X2 + \ldots\ldots + \beta nXn$$

Where, y is a dependent variable and x1, x2 ... and Xn are explanatory variables.

Sigmoid Function:

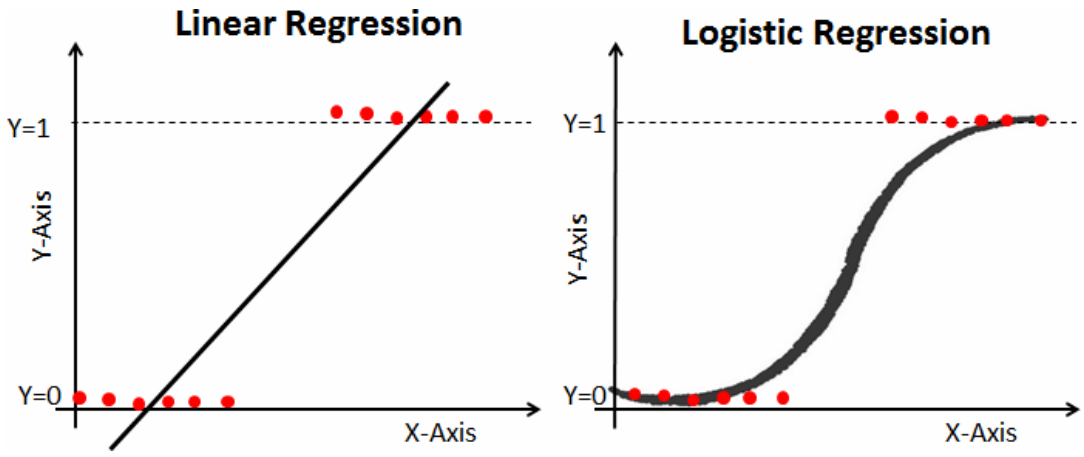
$$p = 1/1 + e^{-y}$$

Apply Sigmoid function on linear regression:

$$p = 1/1 + e^{-(\beta 0 + \beta 1X1 + \beta 2X2 \ldots\ldots \beta nXn)}$$

2. Differentiate between Linear and Logistic Regression

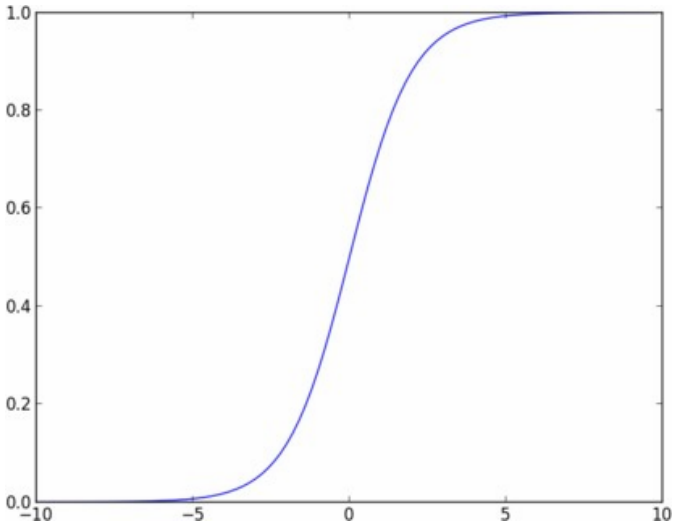
Linear regression gives you a continuous output, but logistic regression provides a constant output. An example of the continuous output is house price and stock price. Example's of the discrete output is predicting whether a patient has cancer or not, predicting whether the customer will churn. Linear regression is estimated using Ordinary Least Squares (OLS) while logistic regression is estimated using Maximum Likelihood Estimation (MLE) approach.



3. Sigmoid Function

The sigmoid function, also called logistic function, gives an 'S' shaped curve that can take any real-valued number and map it into a value between 0 and 1. If the curve goes to positive infinity, y predicted will become 1, and if the curve goes to negative infinity, y predicted will become 0. If the output of the sigmoid function is more than 0.5, we can classify the outcome as 1 or YES, and if it is less than 0.5, we can classify it as 0 or NO. The outputcannotFor example: If the output is 0.75, we can say in terms of probability as: There is a 75 percent chance that a patient will suffer from cancer.

$$f(x) = \frac{1}{1+e^{-(x)}}$$



4. Types of LogisticRegression

Binary Logistic Regression: The target variable has only two possible outcomes such as Spam or Not Spam, Cancer or No Cancer.

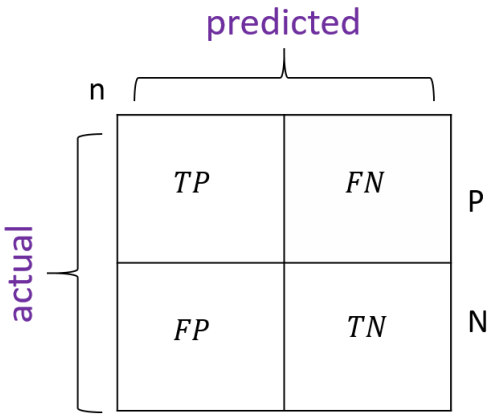
Multinomial Logistic Regression: The target variable has three or more nominal categories such as predicting the type of Wine.

Ordinal Logistic Regression: the target variable has three or more ordinal categories such as restaurant or product rating from 1 to 5.

5. Confusion Matrix Evaluation Metrics

Contingency table or Confusion matrix is often used to measure the performance of classifiers. A confusion matrix contains information about actual and predicted classifications done by a classification system. Performance of such systems is commonly evaluated using the data in the matrix.

The following table shows the confusion matrix for a two class classifier.



Confusion matrix

Here each row indicates the actual classes recorded in the test data set and the each column indicates the classes as predicted by the classifier.
Numbers on the descending diagonal indicate correct predictions, while the ascending diagonal concerns prediction errors.

Some Important measures derived from confusion matrix are:

- **Number of positive (Pos) :** Total number instances which are labelled as positive in a given dataset.
- **Number of negative (Neg)** : Total number instances which are labelled as negative in a given dataset.
- **Number of True Positive (TP) :** Number of instances which are actually labelled as positive and the predicted class by classifier is also positive.
- **Number of True Negative (TN) :** Number of instances which are actually labelled as negative and the predicted class by classifier is also negative.
- **Number of False Positive (FP) :** Number of instances which are actually labelled as negative and the predicted class by classifier is positive.
- **Number of False Negative (FN):** Number of instances which are actually labelled as positiveand the class predicted by the classifier is negative.

- **Accuracy:** Accuracy is calculated as the number of correctly classified instances divided by totalnumber of instances.

  The ideal value of accuracy is 1, and the worst is 0. It is also calculated as the sum of true positive and true negative (TP + TN) divided by the total number of instances.

$$acc = \frac{TP+TN}{TP+FP+TN+FN} = \frac{TP+TN}{Pos+Neg}$$

- **Error Rate:** Error Rate is calculated as the number of incorrectly classified instances divided by total number of instances.

  The ideal value of accuracy is 0, and the worst is 1. It is also calculated as the sum of false positive and false negative (FP + FN) divided by the total number of instances.

$$err = \frac{FP+FN}{TP+FP+TN+FN} = \frac{FP+FN}{Pos+Neg}$$ Or

$$\mathbf{10.}\ err = 1 - acc$$

- **Precision:** It is calculated as the number of correctly classified positive instances divided by the total number of instances which are predicted positive. It is also called confidence value. The ideal value is 1, whereas the worst is 0.

$$precision = \frac{TP}{TP+FP}$$

- **Recall: .**It is calculated as the number of correctly classified positive instances divided by the total number of positive instances. It is also called recall or sensitivity. The ideal value of sensitivity is 1, whereas the worst is 0.

  It is calculated as the number of correctly classified positive instances divided by the total number of positive instances.

$$recall = \frac{TP}{TP+FN}$$

Algorithm (Boston Dataset):

Step 1: Import libraries and create alias for Pandas, Numpy and Matplotlib

Step 2: Import the Social_Media_Adv Dataset

Step 3: Initialize the data frame

Step 4: Perform Data

Preprocessing

- Convert Categorical to Numerical Values if applicable
- Check for Null Value
- Covariance Matrix to select the most promising features
- Divide the dataset into Independent(X) and Dependent(Y)variables.
- Split the dataset into training and testing datasets
- Scale the Features if necessary.

Step 5: Use Logistic regression(Train the Machine) to Create Model

```
# import the class
from sklearn.linear_model import LogisticRegression# instantiate the model
  (using the default parameters)
logreg=LogisticRegression()# fit the model ith data
logreg.fit(xtrain,ytrain)
# y_pred=logreg.predict(xtest)
```

Step 6: Predict the y_pred for all values of train_x and test_x

Step 7: Evaluate the performance of Model for train_y and test_y

Step 8: Calculate the required evaluation parameters

```
from sklearn.metrics import precision_score,confusion_matrix,accuracy_score,recall_score
cm= confusion_matrix(ytest, y_pred)
```

Conclusion:

In this way we have done data analysis using logistic regression for Social Media Adv. and evaluate the performance of model.

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
df=pd.read_csv("Social_Network_Ads.csv")
```

In [6]:
```
df
```

Out[6]:

	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
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

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

Out[10]:
```
np.int64(0)
```

Select Feature and Target

In [13]:
```
X = df[['Age', 'EstimatedSalary']].values
Y = df['Purchased'].values
```

Splitting Dataset

In [14]:
```
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,random_state=50)
```

Applying SMOTE

In [15]:
```
from imblearn.over_sampling import SMOTE

smote=SMOTE(random_state=50)
X_train, Y_train = smote.fit_resample(X_train, Y_train)
```

In [16]:
```
pd.Series(Y_train).value_counts()
```

Out[16]:
```
0    203
1    203
Name: count, dtype: int64
```

Feature Scaling

In [19]:
```
from sklearn.preprocessing import StandardScaler

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

Training the Model

In [38]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

regression = LogisticRegression(max_iter=1500)


param_grid = {
    'C': [0.01, 0.1, 1, 10, 100],  # Inverse of regularization strength
    'penalty': ['l1', 'l2'],       # Type of regularization
    'solver': ['liblinear', 'saga'] # Solvers that support L1 and L2
}

# Perform GridSearchCV
grid_search = GridSearchCV(regression, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, Y_train)

regression=LogisticRegression(C=0.1,penalty='l1',solver='saga')
regression.fit(X_train,Y_train)
```

Out[38]:
```
LogisticRegression(C=0.1, penalty='l1', solver='saga')
```

In [39]:
```
print("Best Parameters:", grid_search.best_params_)
print("Best Score:", grid_search.best_score_)

# Use best model to predict on test set
best_model = grid_search.best_estimator_
test_accuracy = best_model.score(X_test, Y_test)
print("Test Accuracy:", test_accuracy)
```

```
Best Parameters: {'C': 0.1, 'penalty': 'l1', 'solver': 'saga'}
Best Score: 0.8499548328816621
Test Accuracy: 0.825
```

Predictions

In [40]:
```
Y_pred=regression.predict(X_test)
Y_pred
```

Out[40]:
```
array([0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0])
```

Confusion Matrix

In [41]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,precision_score,recall_score

cm=confusion_matrix(Y_test,Y_pred)
TP = cm[1, 1]
FP = cm[0, 1]
TN = cm[0, 0]
FN = cm[1, 0]

accuracy = accuracy_score(Y_test, Y_pred)
error_rate = 1 - accuracy
precision = precision_score(Y_test, Y_pred)
recall = recall_score(Y_test, Y_pred)
```

In [42]:
```
print(f'Confusion Metrix: \n {cm}')
print(f"TP: {TP}, FP: {FP}, TN: {TN}, FN: {FN}")
print(f"Accuracy: {accuracy:.2f}")
print(f"Error Rate: {error_rate:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
```

```
Confusion Metrix:
 [[49  5]
 [ 9 17]]
TP: 17, FP: 5, TN: 49, FN: 9
Accuracy: 0.82
Error Rate: 0.18
Precision: 0.77
Recall: 0.65
```

In [43]:
```
plt.figure(figsize=(7,7))
sns.heatmap(cm,annot=True,fmt='d',cmap='Reds',xticklabels=['Not Purchased', 'Purchased'], yti
```

```
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Metrix")
plt.tight_layout()
plt.show()
```