CSE 4020 - MACHINE LEARNING

Lab 29+30

Lab Task4(Logistic Regression)

Submitted by: Alokam Nikhitha(19BCE2555)

Question:

Train a Logistic regression model to predict the charges is there or not for insurance company from given features.

Dataset Used: "insurance.csv" as provided.

Procedure:

- We first import the dataset into our workspace the use of pandas.
- We then need to determine at the impartial and based attributed for use in our regression version.
- We then need to initialize our Linear regression version and logistic regression and match it to the X and y attributes.
- Next, we need to create any other variable to save the outcomes of X set as anticipated with the aid of using our regression version.
- We then can discover the scatter plot of our units and the exceptional match Regression line.
- Finally, we calculate our assessment metrics to test the accuracy of our version.

Code Snippets and Explanation:

```
In [1]: #Importing the Libraries
        import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
```

Here we are importing the libraries.

```
In [2]: #Importing the Datasets
        dataset = pd.read_csv('insurance.csv')
        X = dataset.iloc[:, 0:1].values
        y = dataset.iloc[:, -1].values
```

We're importing the dataset into our workspace and naming the set of independent attributes X and the set of dependent attributes y.

```
In [3]: #Displaying the dataset
        dataset.head(15)
```

Out[3]:

	age	charges
0	18	0
1	28	0
2	33	1
3	32	0
4	31	0
5	46	1
6	37	1
7	37	1
8	60	1
9	25	0
10	62	1
11	23	0
12	56	1
13	27	0
14	19	0

The first 15 rows of our dataset are seen here. The charges attribute is a categorical attribute, as we can

see, with ‘No' labelled as 0 and ‘Yes' tagged as 1. With values ranging from 18 to 62, the age property is both continuous and discrete.

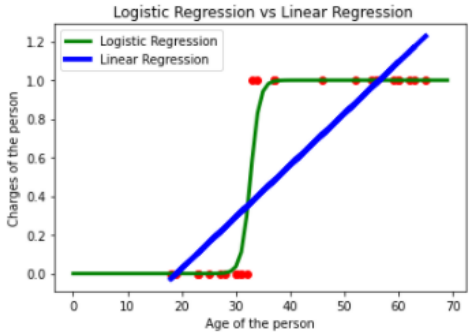
```
In [4]: # Training the Logistic Regression Model
        from sklearn.linear_model import LogisticRegression
        classifier = LogisticRegression(random_state=0)
        classifier.fit(X, y)

Out[4]: LogisticRegression(random_state=0)
```

Here we have trained our Logistic regression model with X and y set.

```
In [5]: # Trainig the Linear Regression model
        from sklearn.linear_model import LinearRegression
        regressor = LinearRegression()
        regressor.fit(X, y)

Out[5]: LinearRegression()
```

Here we have trained our linear regression model with X and y set.

```
In [6]: #Visualizing the Logistic curve and Linear Regressor
        Xs = [i for i in range(0, 70)]
        Ys = [classifier.predict_proba([[value]])[0][1] for value in range(0, 70)]

        plt.scatter(X, y, color='red')
        plt.plot(Xs, Ys, color='Green', linewidth=3, label='Logistic Regression')
        plt.plot(X, regressor.predict(X), color='blue', linewidth=4, label='Linear Regression')
        plt.xlabel('Age of the person')
        plt.ylabel('Charges of the person')
        plt.legend()
        plt.title(' Logistic Regression vs Linear Regression')

Out[6]: Text(0.5, 1.0, ' Logistic Regression vs Linear Regression')
```



The probability line of our logistic regression is presented here. Any value more than 0.5 is defined as "charged,"

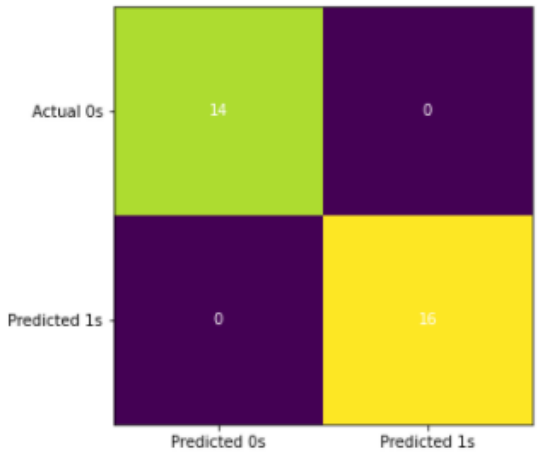
while any value less than 0.5 is classified as "no charge." We have also plotted the values as predicted by linear regressor.

The Logistic Regression is plotted in Green color and Linear Regression in Blue color.

```
In [7]: #Predicting the Test Set Results
        y_pred = classifier.predict(X)
```

We create another vector to store the result of our X set as predicted by the trained classifier.

```
In [8]: #Generating and Visualizing the Confusion Matrix
        from sklearn.metrics import confusion_matrix
        cm = confusion_matrix(y, y_pred)
        fig, ax = plt.subplots(figsize = (5,5))
        ax.imshow(cm)
        ax.grid(False)
        ax.xaxis.set(ticks=(0,1), ticklabels=('Predicted 0s','Predicted 1s'))
        ax.yaxis.set(ticks=(0,1), ticklabels=('Actual 0s', 'Predicted 1s'))
        ax.set_ylim(1.5, -0.5)
        for i in range(2):
            for j in range(2):
                ax.text(j, i, cm[i,j], ha='center', va='center', color='white')
        plt.show()
```



Here, we have visualised our predicted results and actual results. We can clearly see that the true positives and true negatives account for complete results and thus our

logistic regression classifier is 100% accurate, that is, actual set is same as the predicted set.

```
In [9]: #Summarising the classifier's accuracy
        from sklearn.metrics import classification_report
        print(classification_report(y, classifier.predict(X)))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	14
1	1.00	1.00	1.00	16
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

Here we have analysed the performance of our logistic regression classifier.

As we can see, the precision and recall for both 1 and 0 accounts for 1.00, that is all the values of 0 and 1 were identified correctly.

Also, the accuracy of out model is 1.00 (100%) as macro average and weighted average are both 1.00.

```
In [10]: classifier.score(X, y)
Out[10]: 1.0
```

Finally, the classifier score is also 1 as all the values were correctly identified.

Results and Conclusion:

1. Identified ‘no charge’ = 14

2. True ‘no charge’ = 14
3. Identified ‘charged’ = 16
4. True ‘charged’ = 16

5. Precision of ‘no charge’ = 1.00
6. Precision of ‘charged’ = 1.00
7. Recall of ‘no charge’ = 1.00
8. Recall of ‘charged’ = 1.00
9. Model Accuracy = 100%

- **Logistic Regression vs Linear Regression**

```
Out[6]: Text(0.5, 1.0, ' Logistic Regression vs Linear Regression')
```