Department of
Computer Science and Engineering

Title: Linear and Logistic Regression Implementation Using Python

Data Mining LAB

CSE 424

Green University of Bangladesh

1 Objective(s)

- To gather knowledge of the basics of linear and logistic regression.
- To implement linear and logistic regression on real-world datasets.

2 Linear Regression

2.1 Overview of Linear Regression

Linear Regression Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used. This article is going to demonstrate how to use the various Python libraries to implement linear regression on a given dataset. We will demonstrate a binary linear model as this will be easier to visualize.

2.2 Importing All the Required Libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

2.3 Reading the Dataset

```
df = pd.read_csv('bottle.csv')
df_binary = df[['Salnty', 'T_degC']]

# Taking only the selected two attributes from the dataset
df_binary.columns = ['Sal', 'Temp']
#display the first 5 rows
df_binary.head()
```

	Sal	Temp
0	33.440	10.50
1	33.440	10.46
2	33.437	10.46
3	33.420	10.45
4	33.421	10.45

Figure 1: Output

2.4 Exploring the Data Scatter

```
#plotting the Scatter plot to check relationship between Sal and Temp
sns.lmplot(x ="Sal", y ="Temp", data = df_binary, order = 2, ci = None)
```

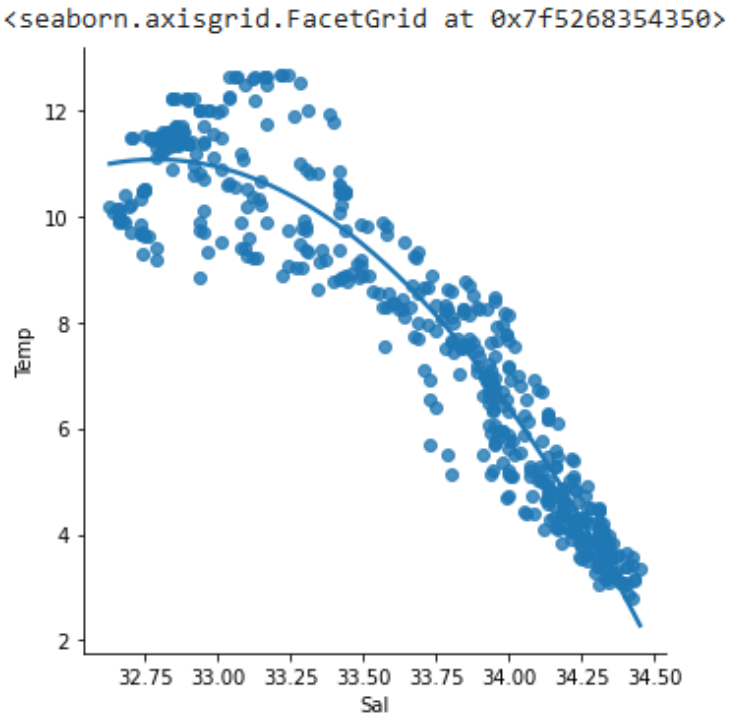

Figure 2: Output

2.5 Data Cleaning

```
# Eliminating NaN or missing input numbers
df_binary.fillna(method ='ffill', inplace = True)
```

2.6 Training the Model

```
X = np.array(df_binary['Sal']).reshape(-1, 1)
y = np.array(df_binary['Temp']).reshape(-1, 1)

# Separating the data into independent and dependent variables
# Converting each dataframe into a numpy array
# since each dataframe contains only one column
df_binary.dropna(inplace = True)

# Dropping any rows with Nan values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)

# Splitting the data into training and testing data
regr = LinearRegression()

regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
```

```
0.8644228954320905
```

Figure 3: Output

2.7 Exploring the Results

```
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color ='b')
plt.plot(X_test, y_pred, color ='k')

plt.show()
# Data scatter of predicted values
```

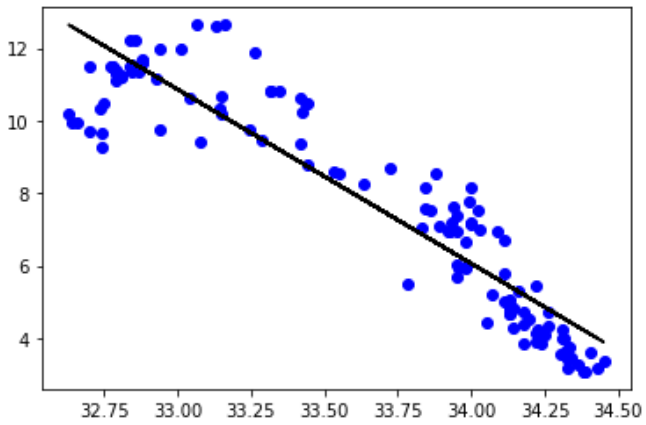

Figure 4: Output

2.8 Evaluation Metrics For Regression

```
from sklearn.metrics import mean_absolute_error,mean_squared_error

mae = mean_absolute_error(y_true=y_test,y_pred=y_pred)
#squared True returns MSE value, False returns RMSE value.
mse = mean_squared_error(y_true=y_test,y_pred=y_pred) #default=True
rmse = mean_squared_error(y_true=y_test,y_pred=y_pred,squared=False)

print("MAE:",mae)
print("MSE:",mse)
print("RMSE:",rmse)
```

```
MAE: 0.8804455681387927
MSE: 1.2283317817479564
RMSE: 1.108301304586418
```

Figure 5: Output

3 Logistic Regression

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.

3.1 Loading Data

```
#import pandas
import pandas as pd
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree',
    'age', 'label']
# load dataset
pima = pd.read_csv("diabetes.csv", header=None, names=col_names)

pima.head()
```

	pregnant	glucose	bp	skin	insulin	bmi	pedigree	age	label
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Figure 6: Output

3.2 Selecting Feature

Here, we need to divide the given columns into two types of variables dependent(or target variable) and independent variable(or feature variables).

```
#split dataset in features and target variable
feature_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']
X = pima[feature_cols] # Features
y = pima.label # Target variable
```

3.3 Splitting Data

To understand model performance, dividing the dataset into a training set and a test set is a good strategy. Let's split dataset by using function train_test_split(). We need to pass 3 parameters features, target, and test_set size. Additionally, we can use random_state to select records randomly.

```
# split X and y into training and testing sets
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_state
    =0)
```

3.4 Model Development and Prediction

First, import the Logistic Regression module and create a Logistic Regression classifier object using LogisticRegression() function. Then, fit the model on the train set using fit() and perform prediction on the test set using predict().

```
# import the class
from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
logreg = LogisticRegression()

# fit the model with data
logreg.fit(X_train,y_train)

#
y_pred=logreg.predict(X_test)
```

3.5 Model Evaluation using Confusion Matrix

A confusion matrix is a table that is used to evaluate the performance of a classification model. One can also visualize the performance of an algorithm. The fundamental of a confusion matrix is the number of correct and incorrect predictions are summed up class-wise.

```
# import the metrics class
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

```
array([[117,  13],
       [ 24,  38]])
```

Figure 7: Output

3.6 Visualizing Confusion Matrix Using Heatmap

Let's visualize the results of the model in the form of a confusion matrix using matplotlib and seaborn. Here, we will visualize the confusion matrix using Heatmap.

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

3.7 Confusion Matrix Evaluation Metrics

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print("Precision:",metrics.precision_score(y_test, y_pred))
print("Recall:",metrics.recall_score(y_test, y_pred))
```

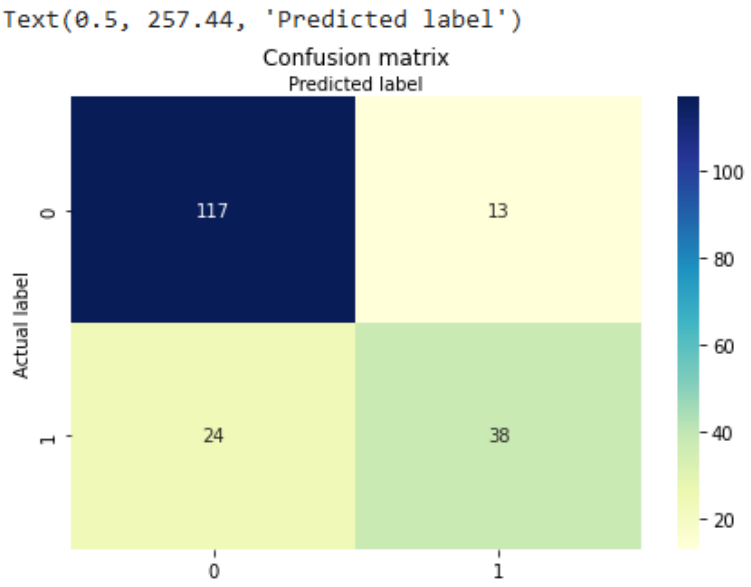

Figure 8: Output

```
Accuracy: 0.8072916666666666
Precision: 0.7450980392156863
Recall: 0.6129032258064516
```

Figure 9: Output

4 Discussion & Conclusion

Based on the focused objective(s) to understand about linear and logistic regression, the additional lab exercise made me more confident towards the fulfilment of the objectives(s).

5 Lab Exercise (Submit as a report)

Develop a jupyter notebook in Colab or Kaggle using all of the codes shown in the class. Choose appropriate datasets from kaggle or any other sources, then show the effect of all codes with proper documentation in the notebook.

6 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.