PMDS505P Data Mining and Machine Learning
Experiment 4

January 2025

1 Work to do today

Note: Make a single pdf file of the work you are doing in jupyter notebook. Upload with proper format. Please mention your name and roll no properly with Experiment number in the first page of your submission.

Binary Classifier: Logistic Regression

Q1. Today we will implement logistic regression to fit a model in connection with the dataset "liver_patient.csv" available for you to download in moodle.

- Download the dataset 'liver_patient.csv' from moodle. This dataset has information regarding whether a person has liver disease or not based on some medical parameters(features) of a person. Open the CSV file and see the different features and the target variable Y also. 1 specifies yes and 0 specifies no in this last column of your dataset, which gives information regarding whether the person has or not liver disease.
- Load the dataset to a dataframe.
- Drop the 'Age','Gender' columns in your dataframe.
- We are looking for a model $h_\theta(\mathbf{X}) = \frac{1}{1+e^{-(\theta_0+\theta_1 X_1+\theta_2 X_2+\ldots+\theta_n X_n)}}$.
- USe MinMaxScaler() to scale the data in the range of 0 to 1.
- Split the data into training and testing sets using appropriate functions. Use a 80:20 split and prepare your x_train,x_test,y_train,y_test data.
- Now, import the inbuilt LogisticRegression class and create an object of this class and fit the model using training data as you have done in the linear and multiple linear regression case like last lab.
  from sklearn.linear_model import LogisticRegression
  logisticR = LogisticRegression()
  logisticR.fit(X_train, y_train)

- Now if you want to see the predictions of your model on the test set you can use **y_pred = logisticR.predict(X_test)**
- to find out the accuracy of your model you can use
  from sklearn.metrics import accuracy_score
  print(accuracy_score(y_test,y_predict))

Q2. Since its difficult to visualize the decision boundary in the above case we will take case where we will generate some dummy data with three features and the respective classes and implement logistic regression and visualize the decision boundary.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

#Let us generate a fake data first
X, y = make_classification(
    n_samples=200,        # Total number of samples in the dataset
    n_features=3,         # Total number of features (columns) for each sample
    n_informative=3,      # Number of informative features truly relevant for classification
    n_redundant = 0,      # can be used for generating new features as linear combination of
                          # exixting features
    n_clusters_per_class=1, # Number of clusters per class
    class_sep=1.5,        # Separation between classes (higher means easier classification)
    random_state=42 )      # Seed for reproducibility
```

Figure 1: Enter Caption

Now we can perform train test split fit the model and get the coefficients to get the decision boundary. Further we will plot the feature vectors and the respective classes and

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Train logistic regression model
logisticR = LogisticRegression()
logisticR.fit(X_train, y_train)

# Predict and calculate accuracy
y_pred = logisticR.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:",accuracy)

# Get model coefficients and intercept
coef = logisticR.coef_[0]
intercept = logisticR.intercept_[0]
print("Coefficients:", coef)
print("Intercept:", intercept)
```

Figure 2: Enter Caption

```python
# Create a 3D plot for the decision boundary
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')

# Plot the data points
ax.scatter(X[y == 0, 0], X[y == 0, 1], X[y == 0, 2], color='blue', label='Class 0', alpha=0.6)
ax.scatter(X[y == 1, 0], X[y == 1, 1], X[y == 1, 2], color='red', label='Class 1', alpha=0.6)

# Create a grid for the decision boundary
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 50), np.linspace(y_min, y_max, 50))
```

Figure 3: Enter Caption

the decision boundary as well. We will do a 3D plotting.

Now we can do the plotting

```python
# Create a grid for the decision boundary
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 50), np.linspace(y_min, y_max, 50))

# Calculate the z values (decision boundary plane)
zz = -(coef[0] * xx + coef[1] * yy + intercept) / coef[2]

# Plot the decision boundary plane
ax.plot_surface(xx, yy, zz, color='green', alpha=0.3, edgecolor='none')

# Set plot labels and title
ax.set_xlabel('Feature 1')
ax.set_ylabel('Feature 2')
ax.set_zlabel('Feature 3')
ax.set_title('3D Logistic Regression Decision Boundary')
ax.legend()
ax.view_init(elev=8, azim=90)  # Adjust elevation and azimuth

plt.show()
```

Figure 4: Enter Caption

Q3. Now use any two features (except the age and gender) from the liver_patient.csv dataset and implement logistic regression to predict whether a person has liver disease or not and in this case find the accuracy of the model and plot the decision boundary.

Q4. Using the liver_patient.csv dataset, drop the features age and gender in that dataframe and write down a gradient descent algorithm to implement the same. find the predictions of your model and the accuracy of the same. compare with Q1 results.