CPE383 Machine Learning: Quiz6

1. 2.5 hours. Modify “Logistic Regression Gradient Descent.ipynb” for Iris classification to use the cost function and Jacobian function and the “BFGS” optimization method in SciPy library to:

a. 20 points. Report the total classification accuracy (score) for the Test data by finding the highest probability class as the output.

```
max_class = []
for i in range(predict_y0_before.shape[0]):
  max_val = max(predict_y0_before[i], predict_y1_before[i], predict_y2_before[i])
  if max_val == predict_y0_before[i]:
    max_class.append(0)
  elif max_val == predict_y1_before[i]:
    max_class.append(1)
  elif max_val == predict_y2_before[i]:
    max_class.append(2)
max_class = np.array(max_class)
print("Class with max probability: \n", max_class)
print("Y test: \n", y_test)
is_correct_match = (max_class == y_test)
print("Is Correct Match: \n", is_correct_match)
correct = ((is_correct_match.sum())/len(y_test))*100
print("Correct percent for logistic class prediction: %5.1f%%"%correct)
```

```
Class with max probability:
 [2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0
 2]
Y test:
 [2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0
 1]
Is Correct Match:
 [ True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True False]
Correct percent for logistic class prediction:  97.4%
```

b. 5 points. Print the confusion matrix for the test data of the 3 classes found by your own implementation of the logistic regressor. You can use Sklearn's library;

```
print(confusion_matrix(y0_test,predict_y0))
print(confusion_matrix(y1_test,predict_y1))
print(confusion_matrix(y2_test,predict_y2))
```

```
[[25  0]
 [ 0 13]]
[[20  2]
 [10  6]]
[[28  1]
 [ 0  9]]
```

c. 5 points. Print the confusion matrix for the test data found by sklearn's logistic regressor.

```
print(confusion_matrix(y_test, y_predict))
```

```
array([[13,  0,  0],
       [ 0, 15,  1],
       [ 0,  0,  9]])
```

d. 10 points. For each mis-classified data sample, show the classification probability for class 0, 1, and 2 (we want to see that the probabilities are hovering around the 0.2 to 0.9 range for these data points).

```
d = 2
np.set_printoptions(precision=d, suppress=True)

miss_y0 = predict_y0_before[predict_y0 != y0_test]
miss_y1 = predict_y1_before[predict_y1 != y1_test]
miss_y2 = predict_y2_before[predict_y2 != y2_test]

print("miss_y0:", miss_y0)
print("miss_y1:", miss_y1)
print("miss_y2:", miss_y2)
```

```
miss_y0: []
miss_y1: [0.64 0.24 0.36 0.89 0.17 0.39 0.49 0.35 0.34 0.4  0.36 0.41]
miss_y2: [0.95]
```

2. 30 points. 3 hours. Using the Kaggle open source dataset to predict Attrition (Will an employee continue to stay with our company?) based on several factors using Logistic Regression. Make sure you show the confusion matrix. In the case of a Yes/No classification like this problem it shows false positives and false negatives. The data is available in:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
```

```
# Read the CSV file
Employee_Attrition = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")

# Drop NA
Employee_Attrition.dropna(inplace=True)

# Extract label
label = Employee_Attrition['Attrition']
Employee_Attrition = Employee_Attrition.drop(['Attrition'], axis=1)

# convert string data to unique integer
le = LabelEncoder()
Employee_Attrition['BusinessTravel'] = le.fit_transform(Employee_Attrition['BusinessTravel'])
Employee_Attrition['Department'] = le.fit_transform(Employee_Attrition['Department'])
Employee_Attrition['EducationField'] = le.fit_transform(Employee_Attrition['EducationField'])
Employee_Attrition['Gender'] = le.fit_transform(Employee_Attrition['Gender'])
Employee_Attrition['JobRole'] = le.fit_transform(Employee_Attrition['JobRole'])
Employee_Attrition['MaritalStatus'] = le.fit_transform(Employee_Attrition['MaritalStatus'])
Employee_Attrition['Over18'] = le.fit_transform(Employee_Attrition['Over18'])
Employee_Attrition['OverTime'] = le.fit_transform(Employee_Attrition['OverTime'])
print(Employee_Attrition.dtypes)

# Standardize the data
scaler = StandardScaler()
Employee_Attrition_scaled = scaler.fit_transform(Employee_Attrition)

# Convert standardized data to pandas dataframe
Spotify_Youtube = pd.DataFrame(Employee_Attrition_scaled, columns=Employee_Attrition.columns)
```

```
Age                       int64
BusinessTravel            int64
DailyRate                 int64
Department                int64
```

```
# split test and train
X_train, X_test, y_train, y_test = train_test_split(Employee_Attrition, label, test_size=0.2)

# create the model
model = LogisticRegression(max_iter=10000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# show accuracy and confusion_matrix
print("accuracy: ", model.score(X_test, y_test))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
accuracy:  0.8673469387755102
Confusion Matrix:
 [[239  11]
 [ 28  16]]
```