Aansh Jha Homework 6

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score,
import matplotlib.pyplot as plt
```

Question 1: Load data and prepare the dataset

```
np.random.seed(1234)

file_path = 'data/nyccrashes_cleaned.feather'
data = pd.read_feather(file_path)

feature_columns = ['LATITUDE', 'LONGITUDE', 'NUMBER OF PERSONS KILLED', 'NUMBER OF PEDESTRIAI
                   'NUMBER OF PEDESTRIANS KILLED', 'NUMBER OF CYCLIST INJURED', 'NUMBER OF C
                   'NUMBER OF MOTORIST INJURED', 'NUMBER OF MOTORIST KILLED']
X_features = data[feature_columns].fillna(0)
y_target = (data['NUMBER OF PERSONS INJURED'] > 0).astype(int)

X_train_data, X_test_data, y_train_data, y_test_data = train_test_split(X_features, y_target
```

Question 2: Fit logistic model and evaluate performance

```
logistic_model = LogisticRegression(max_iter=1000, random_state=1234)
logistic_model.fit(X_train_data, y_train_data)
```

```
y_predicted = logistic_model.predict(X_test_data)

confusion_mat = confusion_matrix(y_test_data, y_predicted)
print("Confusion Matrix:\n", confusion_mat)

f1_value = f1_score(y_test_data, y_predicted)
print(f"F1 Score: {f1_value:.2f}")

true_positive, false_positive, false_negative, true_negative = confusion_mat[1, 1], confusion
print(f"True Positives: {true_positive}, False Positives: {false_positive}, False Negatives:
```

```
Confusion Matrix:
 [[204   0]
 [  8 163]]
F1 Score: 0.98
True Positives: 163, False Positives: 0, False Negatives: 8, True Negatives: 204
```

Question 3: Fit L1-regularized logistic model with cross-validation

```
l1_regularized_model = LogisticRegressionCV(penalty='l1', solver='liblinear', cv=5, scoring=
l1_regularized_model.fit(X_train_data, y_train_data)


best_regularization_C = l1_regularized_model.C_[0]
print(f"Best C value for L1 regularized model: {best_regularization_C}")


y_predicted_l1 = l1_regularized_model.predict(X_test_data)
```

```
Best C value for L1 regularized model: 0.046415888336127774
```

Question 4: Compare performance of the two models

```
metric_names = ["Accuracy", "Precision", "Recall", "F1 Score", "AUC"]

logistic_metrics = [
```

```
    accuracy_score(y_test_data, y_predicted),
    precision_score(y_test_data, y_predicted),
    recall_score(y_test_data, y_predicted),
    f1_score(y_test_data, y_predicted),
    roc_auc_score(y_test_data, logistic_model.predict_proba(X_test_data)[:, 1])
]


l1_regularized_metrics = [
    accuracy_score(y_test_data, y_predicted_l1),
    precision_score(y_test_data, y_predicted_l1),
    recall_score(y_test_data, y_predicted_l1),
    f1_score(y_test_data, y_predicted_l1),
    roc_auc_score(y_test_data, l1_regularized_model.predict_proba(X_test_data)[:, 1])
]

print("Performance Comparison:")
print(f"{'Metric':<12}{'Logistic':<12}{'L1 Regularized':<15}")
for metric, logistic_value, l1_value in zip(metric_names, logistic_metrics, l1_regularized_me
    print(f"{metric:<12}{logistic_value:<12.2f}{l1_value:<15.2f}")
```

```
Performance Comparison:
Metric      Logistic    L1 Regularized
Accuracy    0.98        0.98
Precision   1.00        1.00
Recall      0.95        0.95
F1 Score    0.98        0.98
AUC         0.98        0.98
```