Assignment 4

Fall 2022 5710 Machine Learning: Assignment 4, CS 5710

Name: Pallavi H. Meher
SID: 700727681
Email: pxm76810@ucmo.edu

- Programming elements:
    1. Linear Regression
    2. K-Means Clustering
    3. Data Analysis

1. Apply **Linear Regression** to the provided dataset using underlying steps.
- Import the given "**Salary_Data.csv**"
- Split the data in **train_test** partitions, such that **1/3** of the data is reserved as **test subset**.
- **Train** and **predict** the model.
- Calculate the **mean_squared** error
- Visualize both **train** and **test** data using scatter plot.

```
# Read Data
salary = pd.read_csv("Salary_Data.csv")


# Data Glance
print("\033[1m==> Data Overview \n")
display(salary.head())

# Derive Feature and Target Variables
X = salary[["YearsExperience"]] # Feature
y = salary["Salary"] # Target

# Split data into 1/3 proportion
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

# Model Creation
reg = LinearRegression().fit(X_train,y_train)

print("\033[1m\n==> Linear Regression Model \n")
display(reg)

# Find Coefficient and Intercept for Statical refrence to interpret the model
coeff_df = pd.DataFrame(reg.coef_, X.columns, columns=['Coefficient'])
intercept_df = pd.DataFrame(reg.intercept_, X.columns, columns=['Intercept'])
```

```
print("\033[1m==> Coefficient \n")
display(coeff_df)

print("\n\033[1m==> Intercept \n")
display(intercept_df)


# Evaluate the model
preds = reg.predict(X_test)
mse = mean_squared_error(y_test,preds)

print("\n\033[1m==> Mean Square Error is:",mse,"\n\n")

# Train and Test Data Visualization
plt.scatter(X_train,y_train, c='orange',ec='black',s=55,label="Salary")
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.grid(which='major', color='#CCCCCC', linestyle='--')
plt.grid(which='minor', color='#CCCCCC', linestyle=':')
plt.legend(title="Train Dataset")
plt.title('Train Data Scatter Plot')
plt.show()

plt.scatter(X_test,y_test, c='cyan',ec='b',s=55,label="Salary")
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.grid(which='major', color='#CCCCCC', linestyle='--')
plt.grid(which='minor', color='#CCCCCC', linestyle=':')
plt.legend(title="Test Dataset")
plt.title(' Test Data Scatter Plot')
plt.show()
```

2. Apply **K means** clustering in the dataset provided:

- Remove any **null values by** the **mean**.
- Use the **elbow method** to find a good number of clusters with the **K-Means algorithm**
- Calculate the **silhouette score** for the above clustering

```
# Read Data
clustering = pd.read_csv("K-Mean_Dataset.csv")

# Data Glance
print("\033[1m==> Data Overview \n")
display(clustering.head().T)
```

```
# Check Null
print("\033[1m==> Ckeck Null Values \n")
null_ = clustering.isnull().any()
display(null_)

# Collect Null Columns
print("\033[1m==> Columns having Null Values \n")
null_col = clustering.columns[clustering.isnull().any()].tolist()
display(null_col)

# Get Descriptive Stats
print("\033[1m\n==> Genral Stats \n")
display(clustering.describe().T)

# Fill null values with mean value
clustering =
clustering.fillna(value={'CREDIT_LIMIT':4494.449450,'MINIMUM_PAYMENTS':
8637.0})

# Cross Check null values
print("\033[1m==> Filling and Cross Ckeck Null Values \n")
display(clustering.isnull().any())

# Create Features for K - means Model

FEATURES = clustering[[
    'BALANCE',
    'BALANCE_FREQUENCY',
    'PURCHASES',
    'ONEOFF_PURCHASES',
    'INSTALLMENTS_PURCHASES',
    'CASH_ADVANCE',
    'PURCHASES_FREQUENCY',
    'ONEOFF_PURCHASES_FREQUENCY',
    'PURCHASES_INSTALLMENTS_FREQUENCY',
    'CASH_ADVANCE_FREQUENCY',
    'CASH_ADVANCE_TRX',
    'PURCHASES_TRX',
    'CREDIT_LIMIT',
    'PAYMENTS',
    'MINIMUM_PAYMENTS',
    'PRC_FULL_PAYMENT',
    'TENURE'
]]
```

```
# Model Pre Creation
print("\033[1m==> Silhouette Score \n")
# Sum of squared distances
SSE = []
silhouette_avg = []
for cluster in range(2,15):
    # Elbow Method
    kmeans = KMeans(n_clusters = cluster, init='k-means++')
    preds = kmeans.fit_predict(FEATURES)
    cluster_labels = kmeans.labels_
    SSE.append(kmeans.inertia_)

    # silhouette score
    silhouette_avg.append(silhouette_score(FEATURES, cluster_labels))
    score = silhouette_score(FEATURES, preds)
    print("\033[1mFor\033[0m n_clusters = \033[1m{}\033[0m => \033[1msilhouette
score\033[0m is \033[1m{}\033[0m".format(cluster, score))


# Plot Elbow Methid
frame = pd.DataFrame({'Cluster':range(2,15), 'SSE':SSE})
plt.plot(frame['Cluster'], frame['SSE'], marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.grid(which='major', color='#CCCCCC', linestyle='--')
plt.grid(which='minor', color='#CCCCCC', linestyle=':')
plt.title('Elbow Method For Optimal k')
plt.show()


# Plot silhouette score
plt.plot(frame['Cluster'],silhouette_avg,marker='o')
plt.xlabel('Values of K')
plt.ylabel('Silhouette score')
plt.grid(which='major', color='#CCCCCC', linestyle='--')
plt.grid(which='minor', color='#CCCCCC', linestyle=':')
plt.title('Silhouette analysis For Optimal k')
plt.show()
```

3. Try feature scaling and then apply **K-Means** on the scaled features. Did that improve the **Silhouette score**?
- If **yes,** can you justify why?

Yes, it has **improved the scored** as we are **normalizing** the data and **removing the outliers** aslo it can can be improved further using hyper opt technique and pca to reduce dimension to get more sophisticated predictions.

```
print("\033[1m==> Silhouette Score \n")
# Scale the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(FEATURES)

range_n_clusters = list (range(2,10))
for n_clusters in range_n_clusters:
    clusterer = KMeans(n_clusters=n_clusters)
    preds = clusterer.fit_predict(scaled_data)
    centers = clusterer.cluster_centers_

    score = silhouette_score(scaled_data, preds)
    print("\033[1mFor\033[0m n_clusters = \033[1m{}\033[0m => \033[1msilhouette score\033[0m is \033[1m{}\033[0m".format(n_clusters, score))
```

Appendix:
1. GitHub Repo
2. Video