0907726 Applied Machine Learning (Spring 2024)

Midterm Exam

	رقم التسجيل:	KEY	الاسم:

==========

Instructions: Time **180** min. Open book and notes exam. No internet usage. Please answer all problems in the respective shaded rectangular spaces. There are three problems. Notice that this exam has 3 CSV files that you need to copy to the working directory of your Python project.

==========

P1. The following Python code loads the **World Cities dataset** that has the longitude, latitude, and population of many world cities. Complete this code to achieve the following 4 requirements.

[10 marks]

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
data = pd.read_csv('worldcities.csv')
```

1. Find the number of cities that have missing values and drop the cities that have missing values.

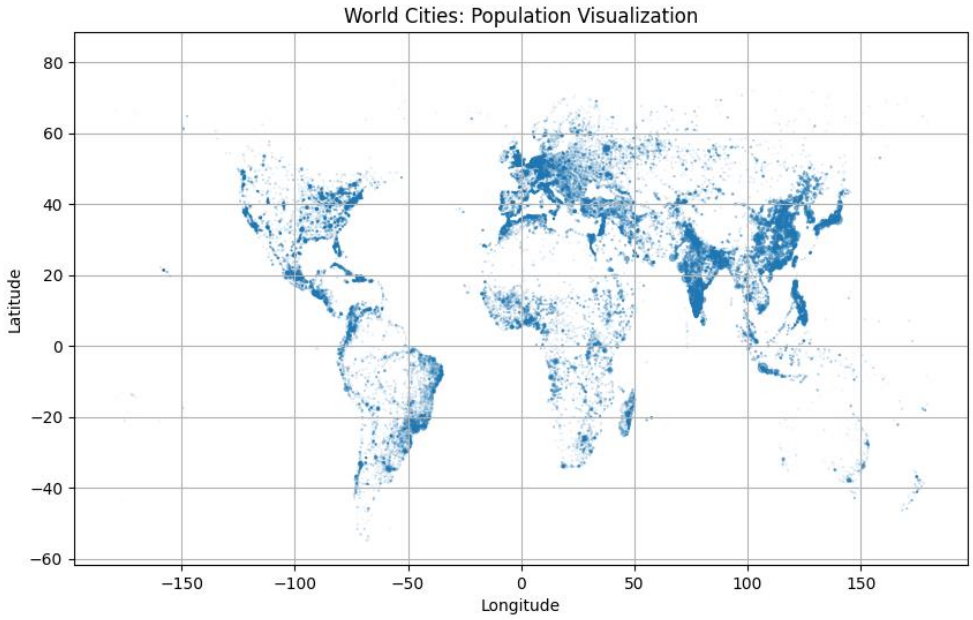
Code	`print(data.info())` `# Remove any rows with missing population or coordinates` `data = data.dropna(subset=['population', 'lat', 'lng'])`
Numbers of cities with missing values	212

2. Convert the population to a suitable size for plotting.

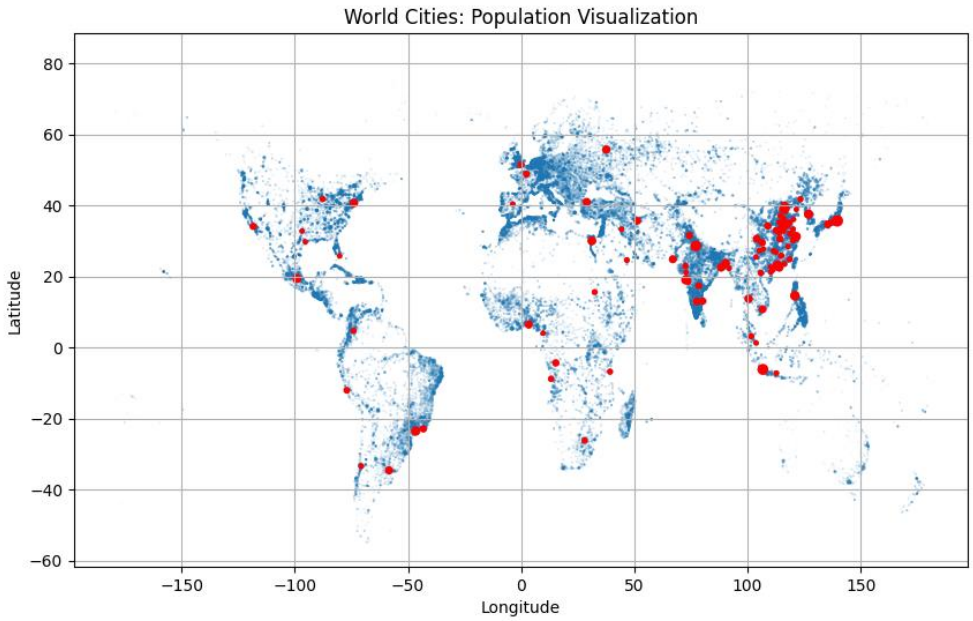
Code	`data['population_size'] = data['population'] / 1000000`

3. Create a scatter plot for this dataset to visualize the city locations and populations. The plot should have suitable title and x and y labels.

Code	`# Create the scatter plot` `plt.figure(figsize=(10, 6))` `plt.scatter(data['lng'], data['lat'],` `  s=data['population_size'], alpha=0.5)` `# Add titles and labels` `plt.title('World Cities: Population Visualization')` `plt.xlabel('Longitude')` `plt.ylabel('Latitude')` `plt.grid(True)` `plt.show()`

Scatter plot	

4. Plot another scatter plot as above but mark the largest 100 cities in red.

Code	``` # Mark the largest 100 cities in red plt.figure(figsize=(10, 6)) plt.scatter(data['lng'], data['lat'], s=data['population_size'], alpha=0.5) data = data[:100] plt.scatter(data['lng'], data['lat'], s=data['population_size'], c='r') # Add titles and labels plt.title('World Cities: Population Visualization') plt.xlabel('Longitude') plt.ylabel('Latitude') plt.grid(True) plt.show() ```
Scatter plot	

P2. The following Python code loads and selects a subset of the **Car MPG dataset**. The objective of this problem is to predict the miles per gallon (mpg) of a vehicle based on characteristics like displacement, horsepower, and weight. Complete this code to achieve the following 5 requirements.

[10 marks]

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Load the dataset
data = pd.read_csv('auto-mpg.csv')
# Select a subset of columns
data = data[['mpg', 'displacement', 'horsepower', 'weight']]
```

1. Perform basic exploration to understand feature types and values.

Code	`print(data.info())` `print(data.describe())`
Are there missing values?	**Yes**
Are there categorical Features?	**No**

2. Perform necessary data preprocessing such as dropping records with missing values and converting data types, if necessary.

Code	`# Drop rows with NaN values` `data.dropna(inplace=True)`

3. Split the dataset to features and response and two subsets: 80% train set and 20% test set.

Code	`# Split the data` `X = data[['displacement', 'horsepower', 'weight']]` `y = data['mpg']` `X_train, X_test, y_train, y_test = train_test_split(X,` `y, test_size=0.2, random_state=42))`

4. Normalize the features using the standard scaler and train the linear regressor on the train subset.

Code	`# Initialize the StandardScaler` `scaler = StandardScaler()` `# Fit on training data and transform both subsets` `X_train_scaled = scaler.fit_transform(X_train)` `X_test_scaled = scaler.transform(X_test)` `# Initialize and train the model` `model = LinearRegression()` `model.fit(X_train_scaled, y_train)`

5. Use root mean squared error (RMSE) to evaluate the model on the test subset.

Code	`# Make predictions` `y_pred = model.predict(X_test_scaled)` `# Evaluate the model` `rmse = np.sqrt(mean_squared_error(y_test, y_pred))` `print(f'Root Mean Squared Error: {rmse:.2f}')`
RMSE	`Root Mean Squared Error: 4.24`

P3. The following Python code loads the **Titanic dataset**. The objective of this problem is to predict whether a passenger survived the Titanic disaster (binary classification: 0 for did not survive, 1 for survived). Complete this code to achieve the following 5 requirements.

[10 marks]

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# Load the dataset
data = pd.read_csv('titanic.csv')
```

1. Perform basic exploration to understand feature types and values.

Code	`print(data.info())`
Which features have missing values?	**Age and Embarked**
Which features are categorical?	**Sex and Embarked**

2. Perform necessary data preprocessing to drop records with missing categorical values and to fill missing numerical values by the median.

Code	`# Drop missing 'Embarked' and fill missing 'Age'` `data.dropna(subset=['Embarked'], inplace=True)` `data['Age'].fillna(data['Age'].median(), inplace=True)`

3. Split the dataset to features and the class and two subsets: 80% train set and 20% test set.

Code	`# Prepare features and target` `X = data[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']]` `y = data['Survived']` `# Split the data` `X_train, X_test, y_train, y_test = train_test_split(` `X, y, test_size=0.2, random_state=42)`

4. Create preprocessing and modeling pipelines. The numerical features should be normalized, the categorical features should be one-hot encoded, and the model should be the Logistic Regression classifier.

Code	`# Create preprocessing and modeling pipeline` `num_feat = ['Pclass', 'Age', 'SibSp', 'Parch', 'Fare']` `cat_feat = ['Sex', 'Embarked']`

```
preprocessor = ColumnTransformer([
    ('num', StandardScaler(),num_feat),
    ('cat',      OneHotEncoder(handle_unknown='ignore'),
cat_feat)
    ])

# Create the pipeline
clf = Pipeline(steps=[('preprocessor', preprocessor),
                      ('classifier',
LogisticRegression(max_iter=500))])
```

5. Train the model on the train set and use the trained model to make predictions for the test set, then evaluate these predictions using the `classification_report()` function.

Code

```
# Train the model
clf.fit(X_train, y_train)

# Make predictions
y_pred = clf.predict(X_test)

# Evaluate the model
print(classification_report(y_test, y_pred))
```

Classification Report

```
              precision    recall  f1-score   support

           0       0.85      0.77      0.81       109
           1       0.68      0.78      0.73        69

    accuracy                           0.78       178
   macro avg       0.77      0.78      0.77       178
weighted avg       0.78      0.78      0.78       178
```

<Good Luck>