The Superior University, Lahore

Assignment-I (Fall 2024)

Course Title:	PAI				Course Code:	CPR601260	Credit Hours: 4
Instructor:					Programme Name:	BSDS	
Semester:	4th	Batch:	F23	Section:	BSDSM-4A	Date:	30 January, 2024
Time Allowed:					Maximum Marks:		
Student's Name:	Fizza Farooq				Reg. No.	Su92-bsdsm-f23-017	

Lab-Task

1: Question
2: Question

Lab 1 Task 1

Task: Kaggle Competition: House Price Prediction

Accuracy Score In kaggle

Code with Outputs And Explanation

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

The Jupyter Notebook imports essential libraries after completing checks on received input files.

Libraries Imported:

- numpy (Numerical computations)
- matplotlib.pyplot & seaborn (Data visualization)
- sklearn.model_selection.train_test_split (Splitting data for training/testing)
- sklearn.ensemble.RandomForestRegressor (Machine learning model)
- sklearn.metrics.mean_absolute_error (Evaluating model performance)

Checking for Available Files in Kaggle Environment:

Through the os.walk('/kaggle/input') protocol the program identifies all accessible files present in the Kaggle dataset directory.

```
data = pd.read_csv("/kaggle/input/home-data-for-ml-course/train.csv")
data.head()
```

Python

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2008
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	5	2007
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	9	2008
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2006
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	12	2008

5 rows × 81 columns

Reads the dataset from a CSV file.Displays the first 5 rows of the dataset using .head().

```
data.head(5)
```

Python

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold	YrSold
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2008
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	5	2007
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	9	2008
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	2	2006
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	12	2008

5 rows × 81 columns

.head(5): Shows the first 5 rows of the dataset.

```
data.tail(5)
```

Python

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	MoSold
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	8
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0	2
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	GdPrv	Shed	2500	5
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	4
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0	6

5 rows × 81 columns

.tail(5): Shows the last 5 rows of the dataset.

Reloading Data

Select target and features

```
path = '../input/home-data-for-ml-course/train.csv'
data = pd.read_csv(path)
```

[5]

This cell re-loads the same dataset from the same file.

```
y = data['SalePrice']
features = ['LotArea', 'YearBuilt', '1stFlrSF', '2ndFlrSF', 'FullBath', 'BedroomAbvGr', 'TotRmsAbvGrd']
X = data[features]
```

- **Target Variable (y)**: The column "SalePrice" is selected as the variable to be predicted (house price).
- **Feature Selection (X)**:
    - The model will use these features to predict house prices:
        - LotArea (Size of the lot in square feet)
        - YearBuilt (Year when the house was built)
        - 1stFlrSF (Size of the first floor in square feet)
        - 2ndFlrSF (Size of the second floor in square feet)
        - FullBath (Number of full bathrooms)
        - BedroomAbvGr (Number of bedrooms above ground level)
        - TotRmsAbvGrd (Total number of rooms above ground level)

Split data into training and validation datasets

```
X_train, X_valid, y_train, y_valid = train_test_split(X, y, train_size=0.8, test_size=0.2, random_state=0)
```

train_test_split function:

- Splits the dataset into training (80%) and validation (20%) sets.
- Ensures randomness with random_state=0, so results are reproducible.

The model is trained on X_train, y_train and tested on X_valid, y_valid to check its performance.

```
model = RandomForestRegressor(random_state=0)
```

- **RandomForestRegressor**:
    - A powerful ensemble learning method that combines multiple decision trees.
    - The random_state=0 ensures reproducibility.

```
model.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(random_state=0)
```

The **.fit()** function trains the RandomForestRegressor model using the training dataset.

```
preds = model.predict(X_valid)

# Calculate the mean absolute error
mae = mean_absolute_error(y_valid, preds)
print(f"Mean Absolute Error: {mae}")
```

```
Mean Absolute Error: 23740.979228636657
```

The trained model predicts house prices for the validation set.

- **Mean Absolute Error (MAE)** is calculated using the predicted values (preds) and the actual values (y_valid).
- **MAE Interpretation:**
  - In this case, **MAE = 23740.97**, meaning that on average, the model's predictions are off by about $23,740 from the actual sale price.
  - A lower MAE is preferred, indicating better model accuracy.

Prepare test data for submission

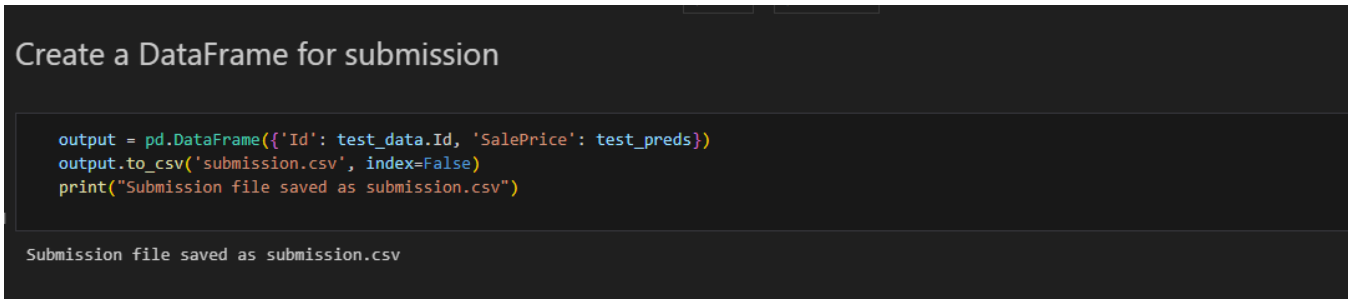
+ Code + Markdown

```
test_data = pd.read_csv('../input/home-data-for-ml-course/test.csv')
X_test = test_data[features]

test_preds = model.predict(X_test)
```

- **Loading the Test Data**:
  - Reads the test dataset (test.csv), which does not have SalePrice values.
  - Only extracts the selected features (features) for prediction.

- **Making Predictions on Test Data**:
    - The trained RandomForestRegressor model predicts house prices for the test dataset.
    - The predictions are stored in test_preds.



Creating Submission File:

- A new DataFrame is created with:
    - Id: The ID of each house from the test dataset.
    - SalePrice: The predicted sale price from test_preds.

Saving the Submission File:

- Writes the DataFrame to a CSV file named submission.csv, which can be submitted to a competition (e.g., Kaggle).

Final Output Message:

- Prints "Submission file saved as submission.csv" to confirm the file was saved successfully.