Name: ABDULRAHMAN ALMYMAN

Student Numbe[redacted]

1.IDE:

Jupyter notebook(easy to use) & Google Collab for working online.

2. libraries used to implement the models:

-Pandas: to read the data easily.

-Sklearn(include: sklearn.preprocessing, sklearn.model_selection, sklearn.metrics, sklearn.ensemble, etc.)

3. Description of the chosen algorithms including all the selected hyperparameters:

SVR() because it has the best score

Selected Hyperparameters:

default hyperparameters

4. Description of Data Preprocessing Applied:

HW3:

- aqardata_2

1- Separate features and target variable

2- Encode categorical variables with LabelEncoder

3- # Initialize the KFold object with shuffle and random_state

class sklearn.model_selection.KFold(, *, shuffle=true, random_state=24)

4- # Initialize a list to store the mean squared errors, mean absolute errors, RMSE, and R2 scores for each model

5- Define the models to evaluate

6 - Iterate over the models and perform cross-validation & Initialize lists to store the evaluation scores for each fold

7 - # Perform cross-validation & Split the data into training and testing sets for this fold

8 - # Fit the model on the training data

9- # Make predictions on the testing data

10 - # Calculate the evaluation scores for this fold

11 - # Append the evaluation scores to the respective lists

12 - Calculate the mean of the evaluation scores for this model
13 - # Append the average evaluation scores to the list
14 - # Sort the models based on their average mean squared error in ascending order
15 - # Get the best performing model and its corresponding evaluation scores
16- # Print the best performing model and its evaluation scores
merging solution(not yet):
-aqardata_2
-DocRealestateSale
-DocRealestateSale2023Q3
-Transactionssaleforrealestate

5. Description of all other steps that have done to get a final result:

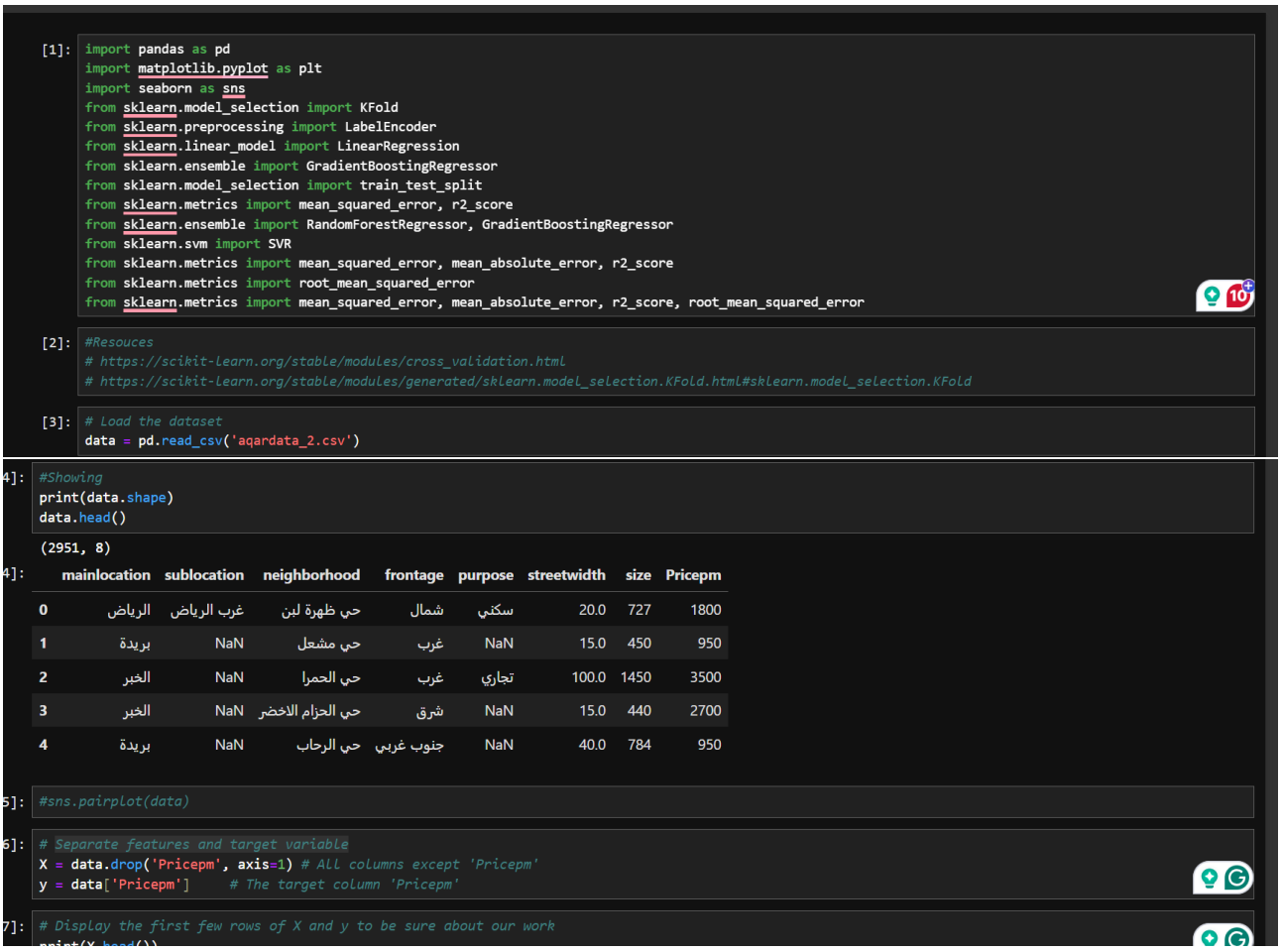
- Selected relevant features for training the model:[mainlocation sublocation neighborhood frontage purpose streetwidth size
-] and make the Target Pricepm
- After the all steps we choose svr because it has highest score.

6. Evaluation results:

The results for HW3:

7. Best Model: Support Vector Regression
8. Mean Squared Error: 7081681.56
9. Mean Absolute Error: 1338.34
10. Root Mean Squared Error: 2661.14
11. R^2 Score: 0.00%

12. Screenshots of all the code & evaluation results from the platform was chosen:



```
[1]: import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.model_selection import KFold
     from sklearn.preprocessing import LabelEncoder
     from sklearn.linear_model import LinearRegression
     from sklearn.ensemble import GradientBoostingRegressor
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import mean_squared_error, r2_score
     from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
     from sklearn.svm import SVR
     from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
     from sklearn.metrics import root_mean_squared_error
     from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score, root_mean_squared_error

[2]: #Resouces
     # https://scikit-learn.org/stable/modules/cross_validation.html
     # https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html#sklearn.model_selection.KFold

[3]: # Load the dataset
     data = pd.read_csv('aqardata_2.csv')

4]: #Showing
    print(data.shape)
    data.head()

    (2951, 8)

4]:
```

	mainlocation	sublocation	neighborhood	frontage	purpose	streetwidth	size	Pricepm
0	الرياض	غرب الرياض	حي ظهرة لبن	شمال	سكني	20.0	727	1800
1	بريدة	NaN	حي مشعل	غرب	NaN	15.0	450	950
2	الخبر	NaN	حي الحمرا	غرب	تجاري	100.0	1450	3500
3	الخبر	NaN	حي الحزام الاخضر	شرق	NaN	15.0	440	2700
4	بريدة	NaN	حي الرحاب	جنوب غربي	NaN	40.0	784	950

```
5]: #sns.pairplot(data)

6]: # Separate features and target variable
    X = data.drop('Pricepm', axis=1) # All columns except 'Pricepm'
    y = data['Pricepm']     # The target column 'Pricepm'

7]: # Display the first few rows of X and y to be sure about our work
```

```
# Separate features and target variable
X = data.drop('Pricepm', axis=1) # All columns except 'Pricepm'
y = data['Pricepm']     # The target column 'Pricepm'
```

```
# Display the first few rows of X and y to be sure about our work
print(X.head())
```

```
  mainlocation sublocation      neighborhood   frontage purpose  streetwidth  \
0         20.0         سكني    شمال  حي ظهرة لبن         غرب الرياض          الرياض
1          بريدة          NaN            غرب          حي مشعل       NaN         15.0
2          الخبر          NaN          100.0          تجاري       غرب          حي الحمرا
3          الخبر          NaN  شرق          حي الحزام الاخضر       NaN         15.0
4          بريدة          NaN          جنوب غربي  حي الرحاب       NaN         40.0

   size
0   727
1   450
2  1450
3   440
4   784
```

```
print(y.head())
```

```
0    1800
1     950
2    3500
4     950
Name: Pricepm, dtype: int64
```

```
# Encode categorical variables with LabelEncoder
encoder = LabelEncoder()
X_encoded = X.apply(encoder.fit_transform)
```

```
# Initialize the KFold object with shuffle and random_state
# class sklearn.model_selection.KFold(n_splits=5, *, shuffle=False, random_state=None)
kf = KFold( shuffle=True, random_state=24)
```

```
# Initialize a list to store the mean squared errors, mean absolute errors, RMSE, and R2 scores for each model
evaluation_scores = []
```

```
# Define the models to evaluate
models = [
    ('Linear Regression', LinearRegression()),
    ('Random Forest', RandomForestRegressor(random_state=35)),
    ('Gradient Boosting', GradientBoostingRegressor(random_state=35)),
    ('Support Vector Regression', SVR())
]
```

```
# Iterate over the models and perform cross-validation
for model_name, model in models:
    # Initialize lists to store the evaluation scores for each fold
    fold_mse_scores = []
    fold_mae_scores = []
    fold_rmse_scores = []
    fold_r2_scores = []
```

```
# Perform cross-validation
for train_index, test_index in kf.split(X_encoded):
    # Split the data into training and testing sets for this fold
    X_train, X_test = X_encoded.iloc[train_index], X_encoded.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
```

```
 # Fit the model on the training data
model.fit(X_train, y_train)
```

```
SVR
SVR()
```

```
# Make predictions on the testing data
y_pred = model.predict(X_test)
```

```
# Calculate the evaluation scores for this fold
fold_mse = mean_squared_error(y_test, y_pred)
fold_mae = mean_absolute_error(y_test, y_pred)
fold_rmse = root_mean_squared_error(y_test, y_pred)  # Use the new function
fold_r2 = r2_score(y_test, y_pred)
```

```
# Append the evaluation scores to the respective lists
fold_mse_scores.append(fold_mse)
fold_mae_scores.append(fold_mae)
fold_rmse_scores.append(fold_rmse)
fold_r2_scores.append(fold_r2)
```

```
# Calculate the mean of the evaluation scores for this model
avg_mse = sum(fold_mse_scores) / len(fold_mse_scores)
avg_mae = sum(fold_mae_scores) / len(fold_mae_scores)
avg_rmse = sum(fold_rmse_scores) / len(fold_rmse_scores)
avg_r2 = sum(fold_r2_scores) / len(fold_r2_scores)
```

```
avg_r2 = sum(fold_r2_scores) / len(fold_r2_scores)

[20]: # Append the average evaluation scores to the list
evaluation_scores.append((model_name, avg_mse, avg_mae, avg_rmse, avg_r2))

[21]: # Sort the models based on their average mean squared error in ascending order
evaluation_scores.sort(key=lambda x: x[1])

[22]: # Get the best performing model and its corresponding evaluation scores
best_model, best_mse, best_mae, best_rmse, best_r2 = evaluation_scores[0]

[24]: # Print the best performing model and its evaluation scores
print(f"Best Model: {best_model}")
print(f"Mean Squared Error: {best_mse:.2f}")
print(f"Mean Absolute Error: {best_mae:.2f}")
print(f"Root Mean Squared Error: {best_rmse:.2f}")
print(f"R² Score: {best_r2:.2f}%")

Best Model: Support Vector Regression
Mean Squared Error: 7081681.56
Mean Absolute Error: 1338.34
Root Mean Squared Error: 2661.14
R² Score: 0.00%
```

THE RESULT FOR merging solution: