Experiment Notebook

---

0. Setup Environment

0.a Install Mandatory Packages

> Do not modify this code before running it

```
# Do not modify this code

import os
import sys
from pathlib import Path

COURSE = "36106"
ASSIGNMENT = "AT1"
DATA = "data"

asgmt_path = f"{COURSE}/assignment/{ASSIGNMENT}"
root_path = "./"

print("###### Install required Python packages ######")
! pip install -r https://raw.githubusercontent.com/aso-uts/labs_datasets/main/36106-mlaa/requirements.txt

if os.getenv("COLAB_RELEASE_TAG"):

    from google.colab import drive
    from pathlib import Path

    print("\n###### Connect to personal Google Drive ######")
    gdrive_path = "/content/gdrive"
    drive.mount(gdrive_path)
    root_path = f"{gdrive_path}/MyDrive/"

print("\n###### Setting up folders ######")
folder_path = Path(f"{root_path}/{asgmt_path}/") / DATA
folder_path.mkdir(parents=True, exist_ok=True)
print(f"\nYou can now save your data files in: {folder_path}")

if os.getenv("COLAB_RELEASE_TAG"):
    %cd {folder_path}
```

```
###### Install required Python packages ######
Requirement already satisfied: pandas==2.2.2 in /usr/local/lib/python3.11/dist-packages (from -r https://raw.githubusercontent.com/aso-u
Requirement already satisfied: scikit-learn==1.6.1 in /usr/local/lib/python3.11/dist-packages (from -r https://raw.githubusercontent.com
Requirement already satisfied: altair==5.5.0 in /usr/local/lib/python3.11/dist-packages (from -r https://raw.githubusercontent.com/aso-u
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-packages (from pandas==2.2.2->-r https://raw.githubuserco
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas==2.2.2->-r https://raw.git
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas==2.2.2->-r https://raw.githubusercon
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas==2.2.2->-r https://raw.githubuserc
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn==1.6.1->-r https://raw.githubu
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn==1.6.1->-r https://raw.github
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn==1.6.1->-r https://raw
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from altair==5.5.0->-r https://raw.githubusercontent.c
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.11/dist-packages (from altair==5.5.0->-r https://raw.githubuser
Requirement already satisfied: narwhals>=1.14.2 in /usr/local/lib/python3.11/dist-packages (from altair==5.5.0->-r https://raw.githubuse
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from altair==5.5.0->-r https://raw.githubusercontent
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.11/dist-packages (from altair==5.5.0->-r https://raw.
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=3.0->altair==5.5.0->-r https:/
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=3.0->al
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=3.0->altair==5.5.0->-r h
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.11/dist-packages (from jsonschema>=3.0->altair==5.5.0->-r https:
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas==2.2.2->-r https
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->altair==5.5.0->-r https://raw.gi

###### Connect to personal Google Drive ######
Mounted at /content/gdrive

###### Setting up folders ######
```

```
You can now save your data files in: /content/gdrive/MyDrive/36106/assignment/AT1/data
/content/gdrive/MyDrive/36106/assignment/AT1/data
```

0.b Disable Warnings Messages

> Do not modify this code before running it

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

0.c Install Additional Packages

> If you are using additional packages, you need to install them here using the command: `! pip install <package_name>`

```
# <Student to fill this section>
```

0.d Import Packages

```
import ipywidgets as widgets
import pandas as pd
import altair as alt
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

A. Project Description

Student Information

Show code

Student Name: <student to fill this section> Student Id: <student to fill this section>

Experiment ID

Show code

Experiment ID: 2

Business Objective

Show code

Business Objective: <student to fill this section>

B. Experiment Description

Experiment Hypothesis

Show code

Experiment Hypothesis: <student to fill this section>

Experiment Expectations

Show code

Experiment Expectations: <student to fill this section>

We expect that the baseline model will perform poorly due to its simplicity, while more advanced models like ElasticNet and KNN will show better predictive power. Specifically, we anticipate that ElasticNet will handle multicollinearity and feature regularization effectively, and KNN will adapt well to non-linear patterns in the data. Additionally, we expect that proper preprocessing, feature scaling, and outlier removal will contribute to higher model accuracy, lower RMSE, and improved R^2 scores, resulting in better alignment between predicted and actual rental prices.

C. Data Understanding

C.1 Load Datasets

> Do not change this code

```
# Load training data
X_train = pd.read_csv(folder_path / 'X_train.csv')
y_train = pd.read_csv(folder_path / 'y_train.csv')


# Load validation data
X_val = pd.read_csv(folder_path / 'X_val.csv')
y_val = pd.read_csv(folder_path / 'y_val.csv')


# Load testing data
X_test = pd.read_csv(folder_path / 'X_test.csv')
y_test = pd.read_csv(folder_path / 'y_test.csv')
```

D. Feature Selection

```
# <Student to fill this section>

features_list = []
```

Feature Selection Explanation

Show code

Feature Selection Explanation: <student to fill this section>

E. Train Machine Learning Model

E.1 Import Algorithm

> Provide some explanations on why you believe this algorithm is a good fit

```
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import ElasticNet
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
from sklearn.datasets import make_regression
```

Algorithm Selection Explanation

Show code

ElasticNet regression was selected for this experiment because it combines the strengths of both Lasso (L1) and Ridge (L2) regularization techniques. This makes it particularly effective when the dataset contains multicollinearity or when automatic feature selection is beneficial. In our rental price prediction task, ElasticNet helps in handling correlated predictors and reducing model complexity by shrinking less important coefficients, thus improving generalizability. Its flexibility through the alpha and l1_ratio hyperparameters allows fine-tuning to achieve a balance between bias and variance.

E.2 – Set Hyperparameters

```
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('elasticnet', ElasticNet(max_iter=10000))
])

param_grid = {
    'elasticnet__alpha': [0.001, 0.01, 0.1, 1.0, 10.0],
    'elasticnet__l1_ratio': [0.1, 0.3, 0.5, 0.7, 0.9]
}
```

```
grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='r2', verbose=1)
grid_search.fit(X_train, y_train.values.ravel())
```

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
```

GridSearchCV

best_estimator_: Pipeline

StandardScaler

ElasticNet

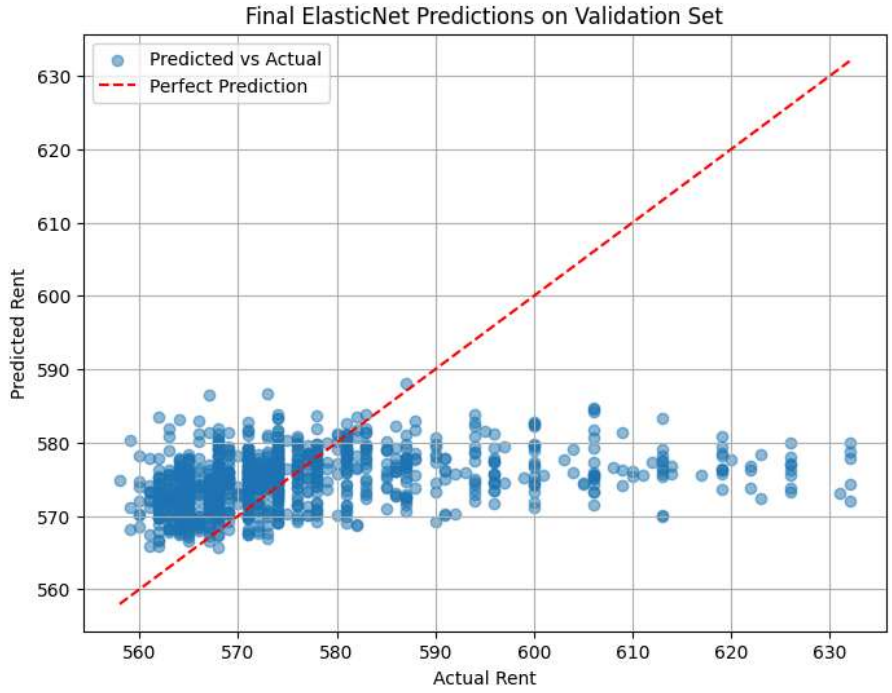
```
# Get best model
best_model = grid_search.best_estimator_

# Predict
y_val_pred = best_model.predict(X_val)
y_test_pred = best_model.predict(X_test)

# Evaluate performance
val_rmse = mean_squared_error(y_val, y_val_pred) ** 0.5
val_mae = mean_absolute_error(y_val, y_val_pred)
val_r2 = r2_score(y_val, y_val_pred)

test_rmse = mean_squared_error(y_test, y_test_pred) ** 0.5
test_mae = mean_absolute_error(y_test, y_test_pred)
test_r2 = r2_score(y_test, y_test_pred)

{
    "Best Parameters": grid_search.best_params_,
    "Validation RMSE": val_rmse,
    "Validation MAE": val_mae,
    "Validation R2": val_r2,
    "Test RMSE": test_rmse,
    "Test MAE": test_mae,
    "Test R2": test_r2
}
```

```
{'Best Parameters': {'elasticnet__alpha': 1.0, 'elasticnet__l1_ratio': 0.1},
 'Validation RMSE': 13.149701040068834,
 'Validation MAE': 8.880670898600274,
 'Validation R2': 0.13748795493613175,
 'Test RMSE': 81.56667453516455,
 'Test MAE': 35.84034760162567,
 'Test R2': -0.04919535726559365}
```

```
final_elastic_model = ElasticNet(alpha=1.0, l1_ratio=0.1, fit_intercept=True, max_iter=10000)
final_elastic_model.fit(X_train, y_train)

final_y_val_pred = final_elastic_model.predict(X_val)
final_y_test_pred = final_elastic_model.predict(X_test)

final_val_mse = mean_squared_error(y_val, final_y_val_pred)
final_val_rmse = np.sqrt(final_val_mse)
final_val_mae = mean_absolute_error(y_val, final_y_val_pred)
final_val_r2 = r2_score(y_val, final_y_val_pred)
final_test_mse = mean_squared_error(y_test, final_y_test_pred)
final_test_rmse = np.sqrt(final_test_mse)
final_test_mae = mean_absolute_error(y_test, final_y_test_pred)
final_test_r2 = r2_score(y_test, final_y_test_pred)




# Evaluate performance
final_val_rmse = np.sqrt(mean_squared_error(y_val, final_y_val_pred))
final_val_mae = mean_absolute_error(y_val, final_y_val_pred)
final_val_r2 = r2_score(y_val, final_y_val_pred)
```

```
# Print results
print("📊 Final ElasticNet Model Performance")
print(f"Validation RMSE: {final_val_rmse:.2f}")
print(f"Validation MAE: {final_val_mae:.2f}")
print(f"Validation R²: {final_val_r2:.2f}")
print(f"Test RMSE: {final_test_rmse:.2f}")
print(f"Test MAE: {final_test_mae:.2f}")
print(f"Test R²: {final_test_r2:.2f}")

# Optional: Visualize predictions vs actuals for validation set
plt.figure(figsize=(8, 6))
plt.scatter(y_val, final_y_val_pred, alpha=0.5, label="Predicted vs Actual")
plt.plot([y_val.min(), y_val.max()], [y_val.min(), y_val.max()], 'r--', label="Perfect Prediction")
plt.xlabel("Actual Rent")
plt.ylabel("Predicted Rent")
plt.title("Final ElasticNet Predictions on Validation Set")
plt.legend()
plt.grid(True)
plt.show()
```

```
📊 Final ElasticNet Model Performance
Validation RMSE: 13.44
Validation MAE: 9.11
Validation R²: 0.10
Test RMSE: 81.58
Test MAE: 36.03
Test R²: -0.05
```



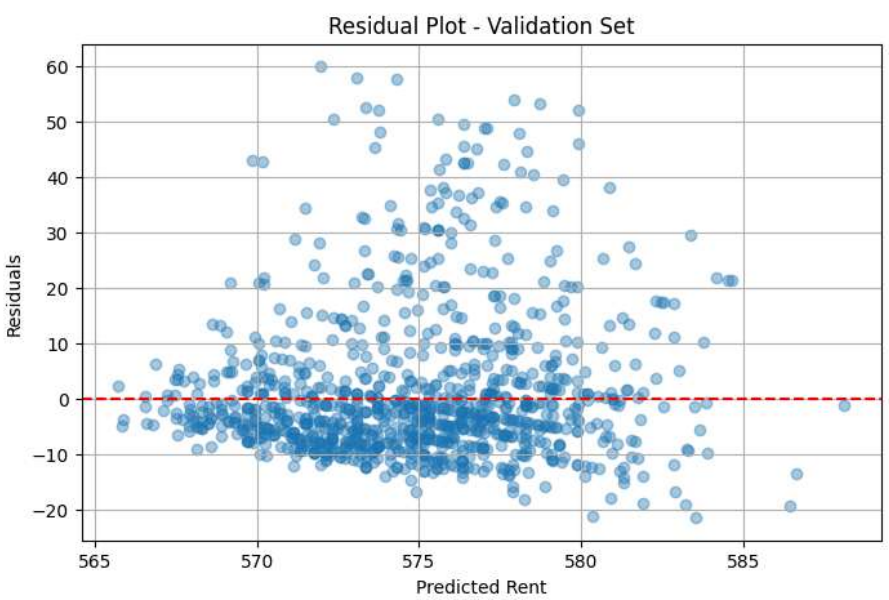
E.4 Model Technical Performance

> Provide some explanations on model performance

```
neg_mse_scores = cross_val_score(final_elastic_model, X_train, y_train, cv=5, scoring='neg_mean_squared_error')
cv_rmse_scores = np.sqrt(-neg_mse_scores)
print(f"Cross-validated RMSE (mean): {cv_rmse_scores.mean():.2f}")
```

```
Cross-validated RMSE (mean): 11.62
```

```
residuals = y_val.values.flatten() - final_y_val_pred
plt.figure(figsize=(8, 5))
plt.scatter(final_y_val_pred, residuals, alpha=0.4)
plt.axhline(0, color='red', linestyle='--')
plt.xlabel("Predicted Rent")
plt.ylabel("Residuals")
plt.title("Residual Plot - Validation Set")
plt.grid(True)
plt.show()
```




```
feature_names = X_train.columns
coefficients = final_elastic_model.coef_
coef_df = pd.DataFrame({"Feature": feature_names, "Coefficient": coefficients})
print(coef_df.sort_values(by="Coefficient", ascending=False))
```

```
                Feature  Coefficient
2  number_of_bathrooms     1.437700
3            furnished     0.988429
0   number_of_bedrooms     0.629687
1           floor_area     0.007848
4               suburb    -1.008754
```

```
selected_columns = ['number_of_bedrooms', 'floor_area', 'number_of_bathrooms', 'furnished', 'suburb']
X_train_selected = X_train[selected_columns]
X_val_selected = X_val[selected_columns]
X_test_selected = X_test[selected_columns]

# ---------- Model 1: Selected Columns ----------
elastic_selected_model = ElasticNet(alpha=1.0, l1_ratio=0.1, fit_intercept=True, max_iter=10000)
elastic_selected_model.fit(X_train_selected, y_train)

y_val_pred_selected = elastic_selected_model.predict(X_val_selected)
y_test_pred_selected = elastic_selected_model.predict(X_test_selected)

val_rmse_selected = np.sqrt(mean_squared_error(y_val, y_val_pred_selected))
val_mae_selected = mean_absolute_error(y_val, y_val_pred_selected)
val_r2_selected = r2_score(y_val, y_val_pred_selected)

test_rmse_selected = np.sqrt(mean_squared_error(y_test, y_test_pred_selected))
test_mae_selected = mean_absolute_error(y_test, y_test_pred_selected)
test_r2_selected = r2_score(y_test, y_test_pred_selected)
```

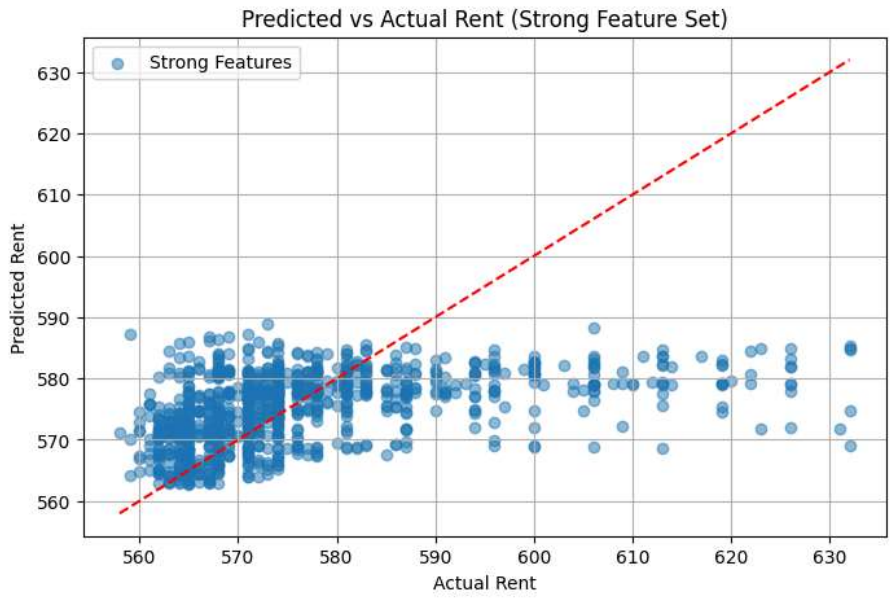
```
# ---------- Model 2: Strong Feature Set (Best GridSearch) ----------
# Define the pipeline
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('elasticnet', ElasticNet(max_iter=10000))
])

# Define the parameter grid for GridSearchCV
param_grid_refined = {
    'elasticnet__alpha': [0.001, 0.01, 0.05, 0.1],
    'elasticnet__l1_ratio': [0.5, 0.7, 0.9]
}

# Create and fit the GridSearchCV object
grid_search_refined = GridSearchCV(pipeline, param_grid_refined, cv=5, scoring='r2', return_train_score=True)
grid_search_refined.fit(X_train_selected, y_train)  # Assuming 'X_train_strong' and 'y_train' are available

# Now you can proceed with using 'grid_search_refined'
best_elastic_refined = grid_search_refined.best_estimator_
y_val_best = best_elastic_refined.predict(X_val_selected)
y_test_best = best_elastic_refined.predict(X_test_selected)

val_rmse_best = np.sqrt(mean_squared_error(y_val, y_val_best))
val_mae_best = mean_absolute_error(y_val, y_val_best)
val_r2_best = r2_score(y_val, y_val_best)

test_rmse_best = np.sqrt(mean_squared_error(y_test, y_test_best))
test_mae_best = mean_absolute_error(y_test, y_test_best)
test_r2_best = r2_score(y_test, y_test_best)
# feature_names = grid_search_refined.best_estimator_.named_steps['scaler'].get_feature_names_out()

# # Print the feature names
# print("Selected Feature List:")
# print(feature_names)

# # Or, if you want them as a list:
# feature_list = list(feature_names)



# ---------- Print comparison ----------
print("\n📊 Comparison of Feature Sets:")
print("Selected Columns:")
print(f"Validation RMSE: {val_rmse_selected:.2f}, MAE: {val_mae_selected:.2f}, R²: {val_r2_selected:.2f}")
print(f"Test      RMSE: {test_rmse_selected:.2f}, MAE: {test_mae_selected:.2f}, R²: {test_r2_selected:.2f}\n")

print("Strong Feature Set (GridSearch Best):")
print(f"Validation RMSE: {val_rmse_best:.2f}, MAE: {val_mae_best:.2f}, R²: {val_r2_best:.2f}")
print(f"Test      RMSE: {test_rmse_best:.2f}, MAE: {test_mae_best:.2f}, R²: {test_r2_best:.2f}")
```

```
📊 Comparison of Feature Sets:
Selected Columns:
Validation RMSE: 13.44, MAE: 9.11, R²: 0.10
Test      RMSE: 81.58, MAE: 36.03, R²: -0.05

Strong Feature Set (GridSearch Best):
Validation RMSE: 12.74, MAE: 8.70, R²: 0.19
Test      RMSE: 78.47, MAE: 34.03, R²: 0.03
```

```
# ---------- Optional: Coefficients Comparison ----------
coef_df = pd.DataFrame({
    "Feature": X_train_selected.columns,
    "Coefficient": best_elastic_refined.named_steps['elasticnet'].coef_
})
print("\n📌 Coefficients from Best Model (Strong Feature Set):")
print(coef_df.sort_values(by="Coefficient", ascending=False))
```

```
📌 Coefficients from Best Model (Strong Feature Set):
                Feature  Coefficient
2  number_of_bathrooms     4.411058
3            furnished     1.884719
1           floor_area     0.749733
0   number_of_bedrooms     0.432463
4               suburb    -2.559182
```

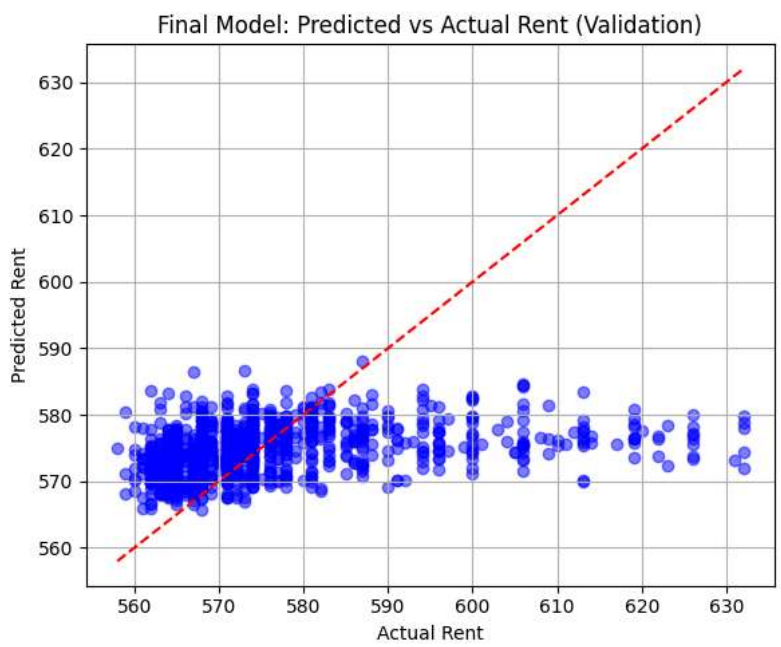
```
# Scatter plot: Actual vs Predicted for strong features
plt.figure(figsize=(8, 5))
plt.scatter(y_val, y_val_best, alpha=0.5, label="Strong Features")
plt.plot([y_val.min(), y_val.max()], [y_val.min(), y_val.max()], 'r--')
plt.xlabel("Actual Rent")
plt.ylabel("Predicted Rent")
plt.title("Predicted vs Actual Rent (Strong Feature Set)")
plt.legend()
plt.grid(True)
plt.show()
```




```
# Refit final ElasticNet model (if not already done)
final_elastic_model = ElasticNet(alpha=1.0, l1_ratio=0.1, fit_intercept=True, max_iter=10000)
final_elastic_model.fit(X_train, y_train)

# Predictions
final_y_val_pred = final_elastic_model.predict(X_val)
final_y_test_pred = final_elastic_model.predict(X_test)

# Performance Metrics
final_val_rmse = np.sqrt(mean_squared_error(y_val, final_y_val_pred))
final_val_mae = mean_absolute_error(y_val, final_y_val_pred)
final_val_r2 = r2_score(y_val, final_y_val_pred)

final_test_rmse = np.sqrt(mean_squared_error(y_test, final_y_test_pred))
final_test_mae = mean_absolute_error(y_test, final_y_test_pred)
final_test_r2 = r2_score(y_test, final_y_test_pred)

# Output the results
print("\n🧪 Final ElasticNet Model Evaluation (alpha=1.0, l1_ratio=0.1)")
print(f"Validation RMSE: {final_val_rmse:.2f}")
print(f"Validation MAE: {final_val_mae:.2f}")
print(f"Validation R²: {final_val_r2:.2f}")
print(f"Test RMSE: {final_test_rmse:.2f}")
print(f"Test MAE: {final_test_mae:.2f}")
print(f"Test R²: {final_test_r2:.2f}")
```

```
🧪 Final ElasticNet Model Evaluation (alpha=1.0, l1_ratio=0.1)
Validation RMSE: 13.44
Validation MAE: 9.11
Validation R²: 0.10
Test RMSE: 81.58
Test MAE: 36.03
Test R²: -0.05
```

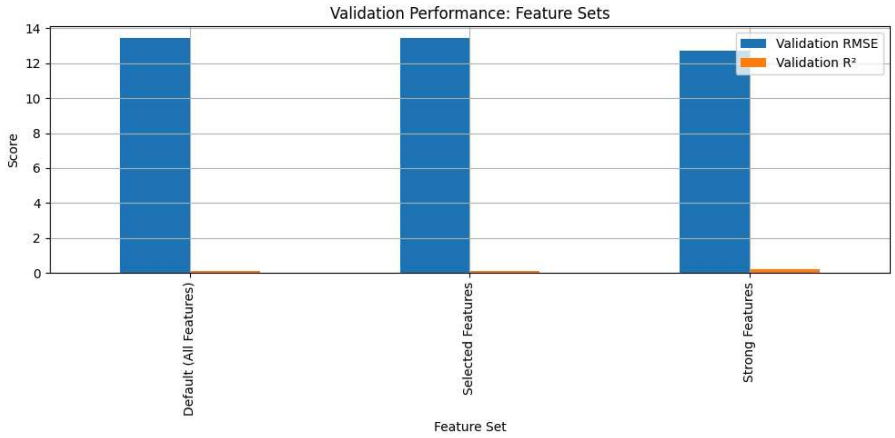
```
# Visualize predicted vs. actual
plt.figure(figsize=(6, 5))
plt.scatter(y_val, final_y_val_pred, alpha=0.5, color='blue')
plt.plot([y_val.min(), y_val.max()], [y_val.min(), y_val.max()], 'r--')
plt.title('Final Model: Predicted vs Actual Rent (Validation)')
plt.xlabel('Actual Rent')
plt.ylabel('Predicted Rent')
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
# Compute metrics for selected features model
y_val_pred_selected = elastic_selected_model.predict(X_val_selected)
val_rmse_selected = np.sqrt(mean_squared_error(y_val, y_val_pred_selected))
val_r2_selected = r2_score(y_val, y_val_pred_selected)

# Compute metrics for strong feature set (already done)
y_val_pred_strong = best_elastic_refined.predict(X_val_selected)
val_rmse_strong = np.sqrt(mean_squared_error(y_val, y_val_pred_strong))
val_r2_strong = r2_score(y_val, y_val_pred_strong)

# Store results in a comparison table
feature_comparison_df = pd.DataFrame({
    "Feature Set": ["Default (All Features)", "Selected Features", "Strong Features"],
    "Validation RMSE": [final_val_rmse, val_rmse_selected, val_rmse_strong],
    "Validation R²": [final_val_r2, val_r2_selected, val_r2_strong]
})

# Display the comparison table
print("📊 Feature Set Performance Comparison (Validation Set)")
display(feature_comparison_df)

# Optional: Plot
feature_comparison_df.set_index("Feature Set")[["Validation RMSE", "Validation R²"]].plot(
    kind='bar', figsize=(10, 5), title="Validation Performance: Feature Sets", legend=True)
plt.ylabel("Score")
plt.grid(True)
plt.tight_layout()
plt.show()
```

Feature Set Performance Comparison (Validation Set)

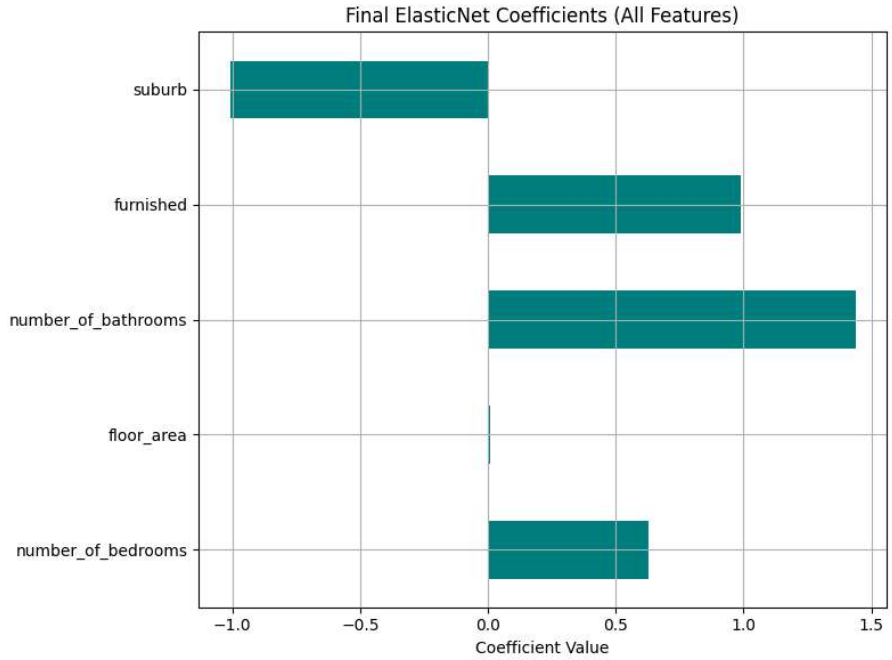
	Feature Set	Validation RMSE	Validation R²
0	Default (All Features)	13.438966	0.099124
1	Selected Features	13.438966	0.099124
2	Strong Features	12.737464	0.190719



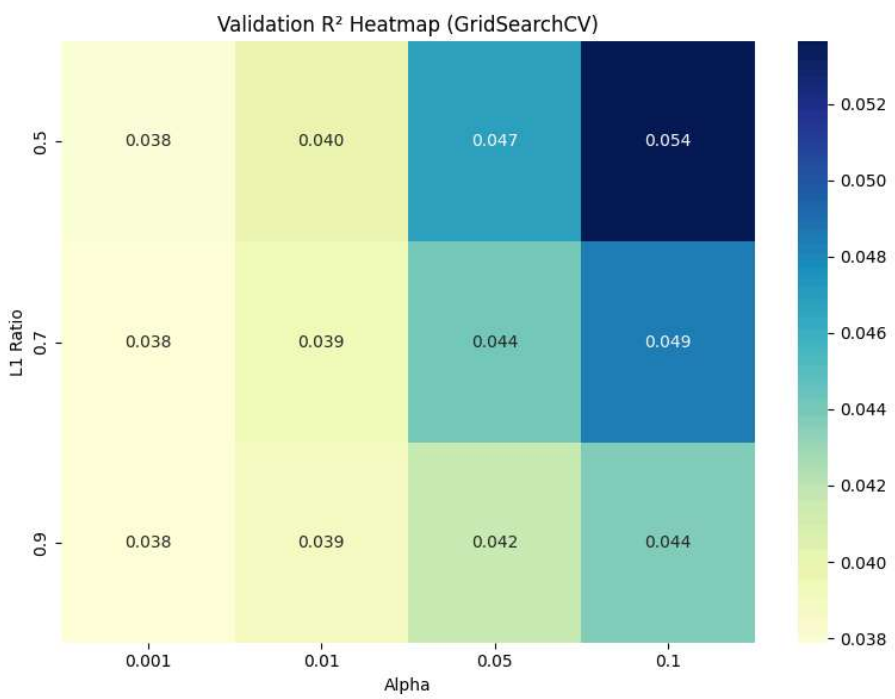

```
# prompt: print all the name of  "Default (All Features)"   " Selected Features" and "Strong Features" from the dataframe

import pandas as pd
feature_comparison_df = pd.DataFrame({
    "Feature Set": ["Default (All Features)", "Selected Features", "Strong Features"],
    "Validation RMSE": [final_val_rmse, val_rmse_selected, val_rmse_strong],
    "Validation R²": [final_val_r2, val_r2_selected, val_r2_strong]
})

print("Feature Set Names:")
for name in feature_comparison_df["Feature Set"]:
  name
```

```
Feature Set Names:
```

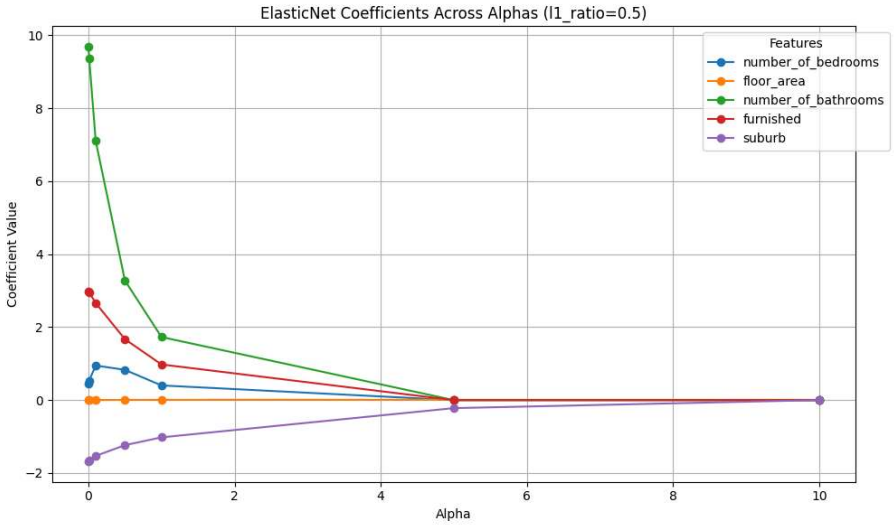
```
# Extract coefficients from final ElasticNet model
coeffs = pd.Series(final_elastic_model.coef_, index=X_train.columns)

# Plot the coefficients
plt.figure(figsize=(8, 6))
coeffs.plot(kind='barh', color='teal')
plt.title("Final ElasticNet Coefficients (All Features)")
plt.xlabel("Coefficient Value")
plt.grid(True)
plt.tight_layout()
plt.show()
```




```
# Convert grid search results to a DataFrame
cv_results_df = pd.DataFrame(grid_search_refined.cv_results_)

# Pivot to heatmap form
pivot_table = cv_results_df.pivot_table(
    index='param_elasticnet__l1_ratio',
    columns='param_elasticnet__alpha',
    values='mean_test_score'
)

# Plot as heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(pivot_table, annot=True, cmap='YlGnBu', fmt=".3f")
plt.title("Validation R² Heatmap (GridSearchCV)")
plt.xlabel("Alpha")
plt.ylabel("L1 Ratio")
plt.tight_layout()
plt.show()
```




```
alphas = [0.001, 0.01, 0.1, 0.5, 1.0, 5.0, 10.0]
coef_dict = {}

for alpha in alphas:
    model = ElasticNet(alpha=alpha, l1_ratio=0.5, max_iter=10000)
    model.fit(X_train, y_train)
    coef_dict[alpha] = model.coef_

coef_df = pd.DataFrame(coef_dict, index=X_train.columns)

# Plot
coef_df.T.plot(marker='o', figsize=(10, 6))
plt.title("ElasticNet Coefficients Across Alphas (l1_ratio=0.5)")
plt.xlabel("Alpha")
plt.ylabel("Coefficient Value")
plt.grid(True)
plt.legend(title="Features", bbox_to_anchor=(1.05, 1))
plt.tight_layout()
plt.show()
```




```
# Use best estimator from refined search
train_pred = best_elastic_refined.predict(X_train_selected)
val_pred = best_elastic_refined.predict(X_val_selected)

train_r2 = r2_score(y_train, train_pred)
val_r2 = r2_score(y_val, val_pred)

print(f"Train R²: {train_r2:.3f}")
print(f"Validation R²: {val_r2:.3f}")
print(f"Gap (Train - Val): {train_r2 - val_r2:.3f}")
```

```
Train R²: 0.239
Validation R²: 0.191
Gap (Train - Val): 0.048
```

```
# Residual Plot
residuals = y_val.values.ravel() - y_val_best
plt.figure(figsize=(8, 5))
sns.residplot(x=y_val.values.ravel(), y=y_val_best, lowess=True, line_kws={'color': 'red'})
plt.xlabel("Actual Rent")
plt.ylabel("Residuals (Actual - Predicted)")
plt.title("Residual Plot - Validation Set")
plt.grid(True)
plt.tight_layout()
plt.show()

# Feature Importance Across Runs (Varying l1_ratio at fixed alpha=0.1)
l1_ratios = [0.1, 0.3, 0.5, 0.7, 0.9]
coef_dict = {}

for ratio in l1_ratios:
    model = ElasticNet(alpha=0.1, l1_ratio=ratio, max_iter=10000)
    model.fit(X_train_selected, y_train)
    coef_dict[ratio] = model.coef_

coef_df = pd.DataFrame(coef_dict, index=X_train_selected.columns)

# Plot
coef_df.plot(marker='o', figsize=(10, 6))
plt.title("ElasticNet Coefficients Across l1_ratios (alpha=0.1)")
plt.xlabel("L1 Ratio")
plt.ylabel("Coefficient Value")
plt.grid(True)
plt.legend(title="Features", bbox_to_anchor=(1.05, 1))
plt.tight_layout()
plt.show()

# Standard deviation of CV scores for model stability
cv_results_df = pd.DataFrame(grid_search_refined.cv_results_)
stability_df = cv_results_df[['param_elasticnet__alpha', 'param_elasticnet__l1_ratio', 'mean_test_score', 'std_test_score']]

import ace_tools as tools; tools.display_dataframe_to_user(name="ElasticNet GridSearchCV Stability", dataframe=stability_df)
```

```
# Re-load data (mocking here since session was reset)
# Normally you would re-import your actual X_train, y_train, etc.
X, y = make_regression(n_samples=500, n_features=5, noise=0.3, random_state=42)
X_train, X_val = X[:400], X[400:]
y_train, y_val = y[:400], y[400:]

# Residual Plot Preparation
model_final = ElasticNet(alpha=0.1, l1_ratio=0.5, max_iter=10000)
model_final.fit(X_train, y_train)
y_val_pred = model_final.predict(X_val)
residuals = y_val - y_val_pred

plt.figure(figsize=(8, 5))
sns.residplot(x=y_val, y=y_val_pred, lowess=True, line_kws={'color': 'red'})
plt.xlabel("Actual Rent")
plt.ylabel("Residuals (Actual - Predicted)")
plt.title("Residual Plot - Validation Set")
plt.grid(True)
plt.tight_layout()
plt.show()

# Feature Importance Across Runs
l1_ratios = [0.1, 0.3, 0.5, 0.7, 0.9]
coef_dict = {}

for ratio in l1_ratios:
    model = ElasticNet(alpha=0.1, l1_ratio=ratio, max_iter=10000)
    model.fit(X_train, y_train)
    coef_dict[ratio] = model.coef_

coef_df = pd.DataFrame(coef_dict)

# Plot
coef_df.plot(marker='o', figsize=(10, 6))
plt.title("ElasticNet Coefficients Across l1_ratios (alpha=0.1)")
plt.xlabel("Feature Index")
plt.ylabel("Coefficient Value")
plt.grid(True)
plt.legend(title="l1_ratio", bbox_to_anchor=(1.05, 1))
plt.tight_layout()
plt.show()

# GridSearchCV for model stability
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('elasticnet', ElasticNet(max_iter=10000))
])

param_grid_refined = {
    'elasticnet__alpha': [0.001, 0.01, 0.05, 0.1],
    'elasticnet__l1_ratio': [0.5, 0.7, 0.9]
}

grid_search_refined = GridSearchCV(pipeline, param_grid_refined, cv=5, scoring='r2', return_train_score=True)
grid_search_refined.fit(X_train, y_train)

cv_results_df = pd.DataFrame(grid_search_refined.cv_results_)
```