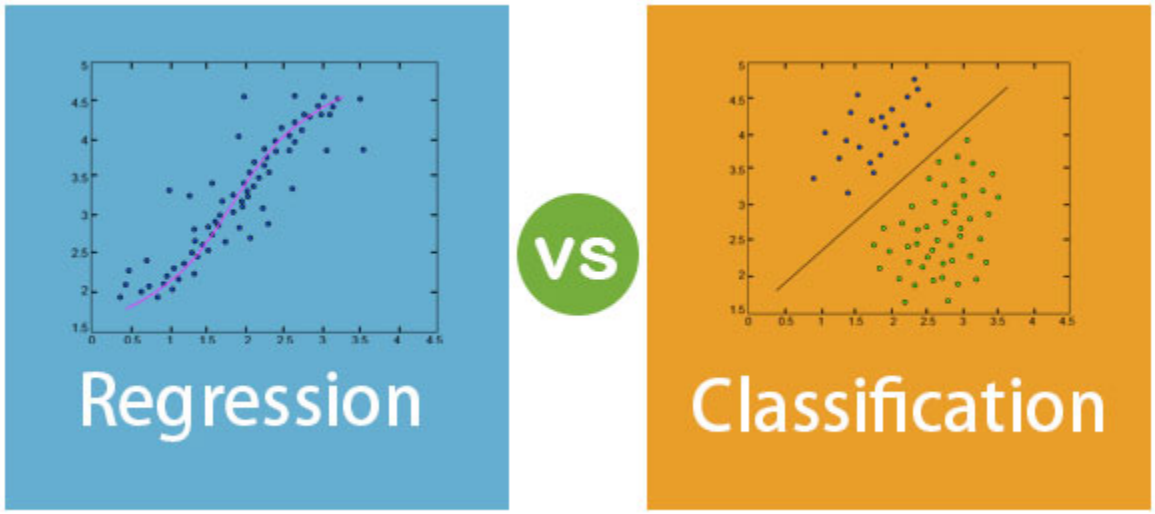
جامعة الإسكندرية
ALEXANDRIA UNIVERSITY
كلية الهندسة
Faculty of Engineering

Machine Learning

Assignment 1:

Classification and Regression



Under the supervision of:

- **Dr. Marwan Torki**

Names:	IDs:
Saad El Dine Ahmed Saad	7370
Morougue Mahmoud Ghazal	7524

Classification

✓Classification of MAGIC Gamma Telescope Dataset using K-Nearest Neighbors (K-NN):

Introduction:

In this analysis, we'll be using the K-Nearest Neighbors (K-NN) classifier to help classify the MAGIC gamma telescope dataset. The dataset consists of two classes: gammas (signal) and hadrons (background).

The primary objectives are:

- Address class imbalance.
- Split the data into training, validation, and testing sets.
- Apply the K-NN classifier with different hyperparameter values (k).
- Evaluate model performance.

Dataset Overview:

The dataset contains a total of 19,020 records with 12,332 gamma events and 6,688 hadron events. The class imbalance issue is addressed by randomly selecting a subset of gamma events to create a balanced dataset.

Data Preprocessing:

- Start by loading the data into a Pandas DataFrame.
- Split the dataset into gamma and hadron classes.
- Select a random subset of gamma events, to produce a balanced dataset, to address class imbalance due to unequal events.

Data Splitting:

Split the balanced dataset is into training (70%), validation (15%), and testing (15%) sets using the "**train_test_split"** function from scikit-learn.

Applying K-NN Classifier with Varying K Values:

Apply K-NN classifier to the training set while varying the hyperparameter k. to explore a range of k values (1, 3, 5, ..., 1000). For each k value, the following **metrics** are calculated:
Accuracy, precision, recall, and F1-score.

Best Model – Accuracy (Results):

The results of applying the K-NN classifier with different K values are as follows:
(considering that the results vary each sample run because the data are shuffled)

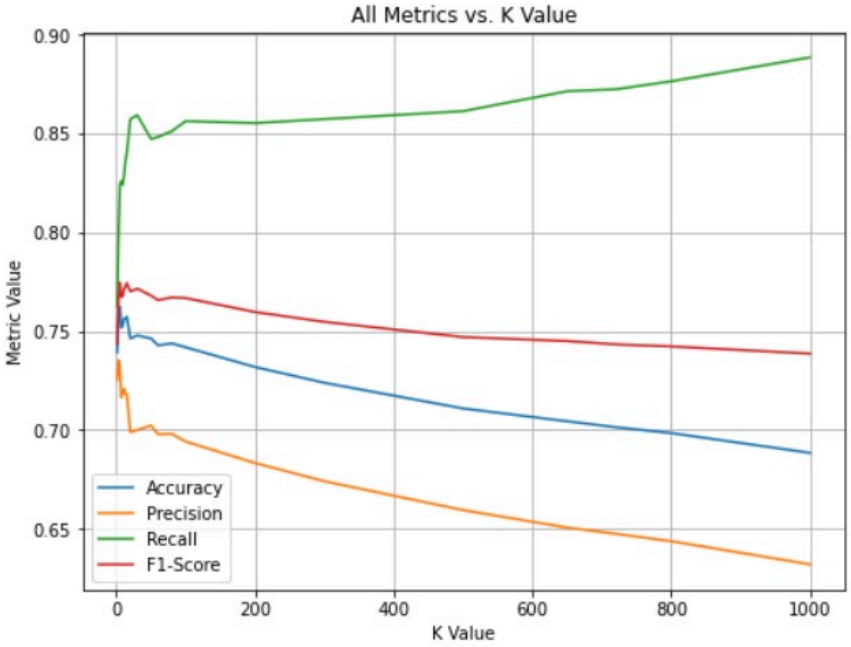
- ✓ **For k = 1:**
  - **Accuracy** is 73.93%
  - **Precision** is 72.54%
  - **Recall** is 76.26%
  - **F1-score** is 74.35%

- ✓ The **highest accuracy** of **76.22%** is achieved for **k = 5**, with a **precision** of 73.06%, **recall** of 82.39%, and **F1-score** of 77.45%.

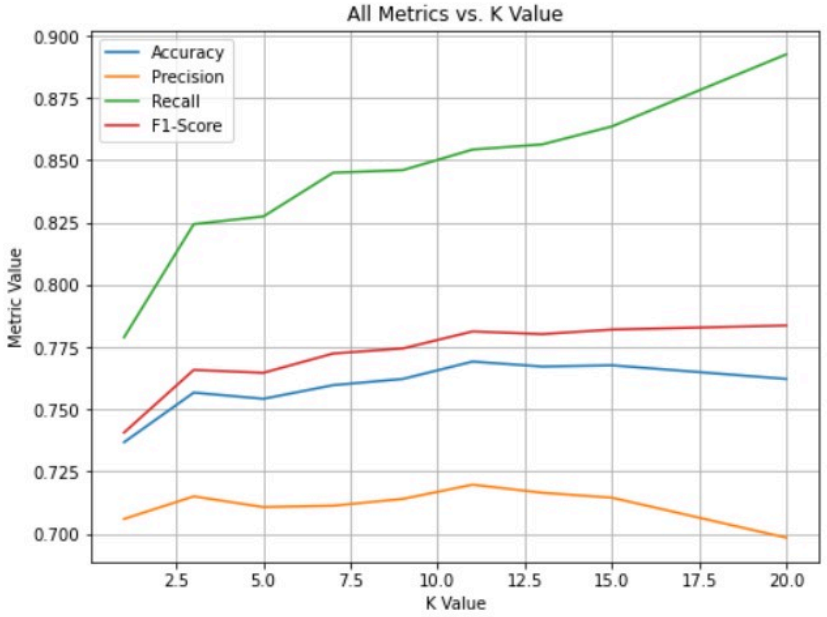
Comparing Models:

The performance metrics for different K values are compared using visualizations, presenting plots of accuracy, precision, recall, and F1-score against k values.

The model with k = 5 achieved the best overall performance according to accuracy and F1-score.



Another sample run where **the model with k = 11** achieved the best overall performance according to accuracy and F1-score.

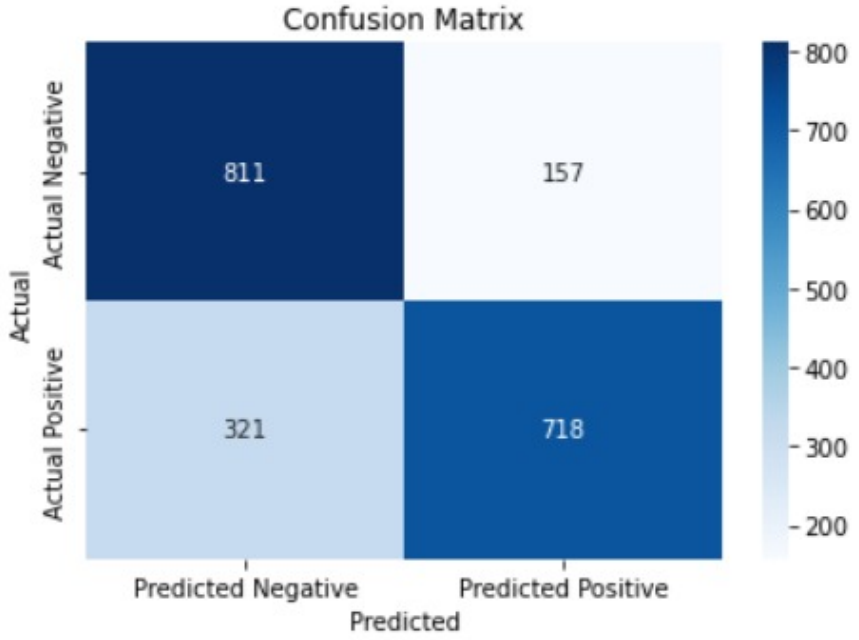


Confusion Matrix (Best Model - Accuracy):

The confusion matrix for the best model (k = 5) based on accuracy is as follows:

- True Positives (TP): **718**
- True Negatives (TN): **811**
- False Positives (FP): **157**
- False Negatives (FN): **321**

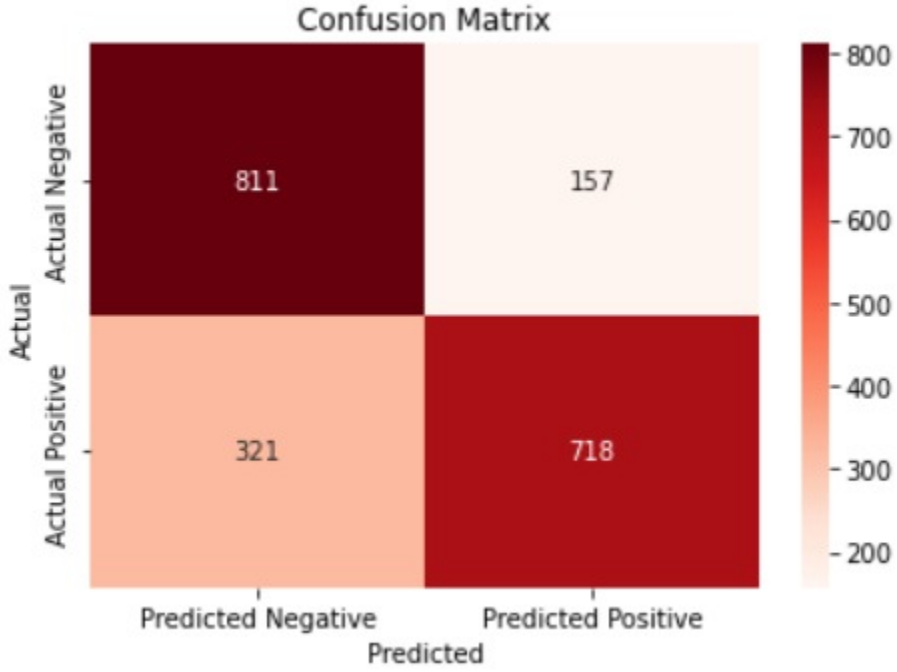
The visualization of the confusion matrix is shown, demonstrating the model's ability to correctly classify gamma and hadron events.



Confusion Matrix (Best Model - F1 Score):

The confusion matrix for the best model (k = 5) based on F1-score is the same as the described above:

- True Positives (TP): **718**
- True Negatives (TN): **811**
- False Positives (FP): **157**
- False Negatives (FN): **321**



Conclusion:

In this analysis of the MAGIC gamma telescope dataset, the K-NN classifier was applied with varying k values to classify gamma and hadron events. The best model, based on both accuracy and F1-score, was achieved with k = 5, showing the highest overall performance. The confusion matrix provides insights into the model's ability to correctly classify events.

This report summarizes the analysis of the MAGIC gamma telescope dataset, showcasing the classification performance of the K-NN classifier and highlighting the importance of addressing class imbalance and selecting an appropriate k value for the model.

Regression

Predicting Median House Values in California: A Regression Analysis

Housing prices are a critical aspect of the real estate market, impacting both homeowners and potential buyers, understanding the factors that influence house prices is essential for making **informed investment decisions**.

In this study, we leverage machine learning techniques to predict median house values based on a dataset containing various attributes related to California houses. We aim to predict the **median house values in California** using **linear**, **lasso**, and **ridge regression models**.

Introduction:

This report details the **application** of **linear regression**, **lasso regression**, and **ridge regression** models to predict **median house values** in California. The analysis includes **data preprocessing**, **model training**, **evaluation**, and **comparison** of the **three regression** models. The primary goal is to **assess the models' performance** in terms of Mean Squared Error (**MSE**) and Mean Absolute Error (**MAE**).

Data Preprocessing:

1. Identifying Skewed Features:

Skewness was calculated for all numeric features in the dataset, and features with skewness greater than 0.5 were identified.

As skewed features may lead to biased models, they were processed to reduce skewness.

2. Log Transformation:

Apply Log transformation to the skewed features to mitigate skewness and make the data more suitable for regression analysis.

This enhances the models' performance by making the data conform more closely to a normal distribution.

3. Z-Score Normalization:

Perform Standardization, or Z-score normalization on the entire dataset, including both features and the target variable.

This step ensures that all variables have a similar scale, which is important for regression models' convergence and performance.

4. Handling Missing Values:

It was observed that the dataset had no missing values, simplifying the data preprocessing process. In cases where missing values exist, appropriate strategies, such as imputation, should be applied.

5. Feature Engineering:

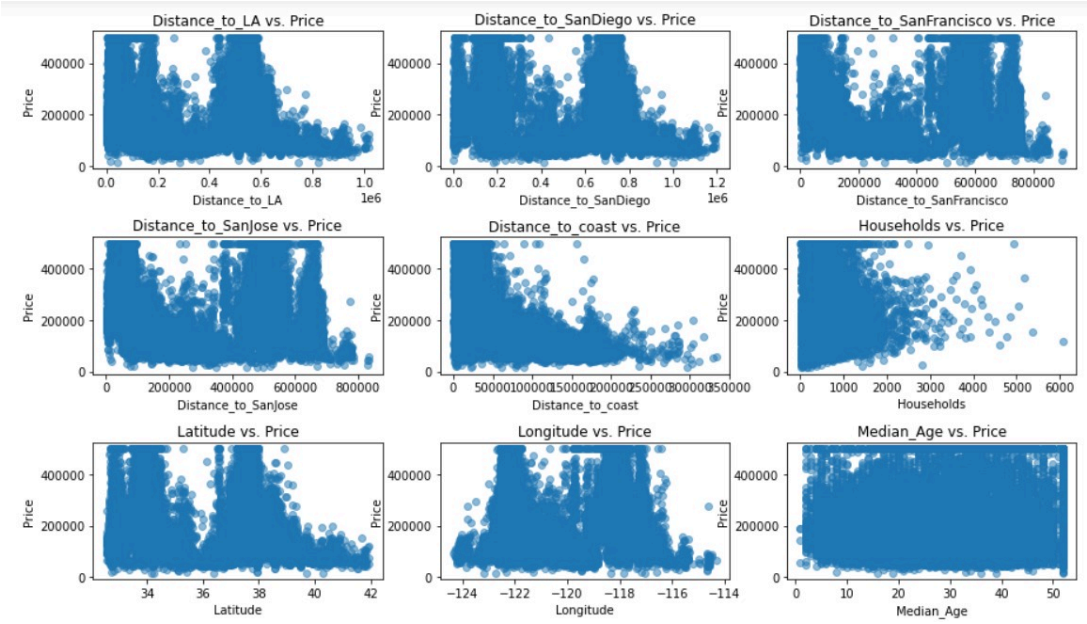
The dataset's features were analyzed for correlations with the target variable (Median House Value). The feature “Distance_to_LA” exhibited the highest positive correlation, followed by “Median_Age”.

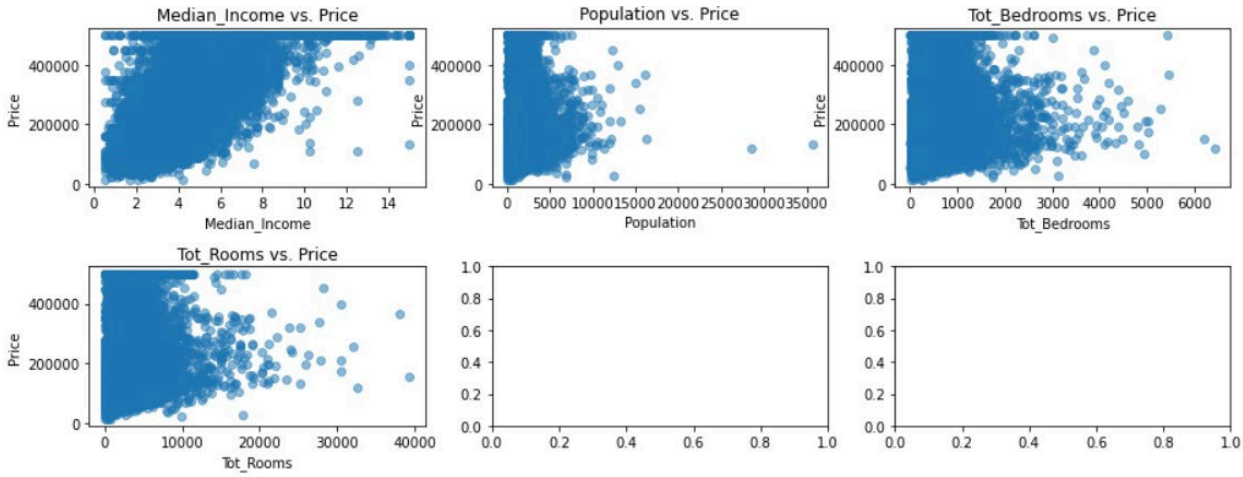
Accordingly, we created **a new feature**, “Distance_Age” by combining both features mentioned above, for further enhancing the predictive power of the models.

Visualization:

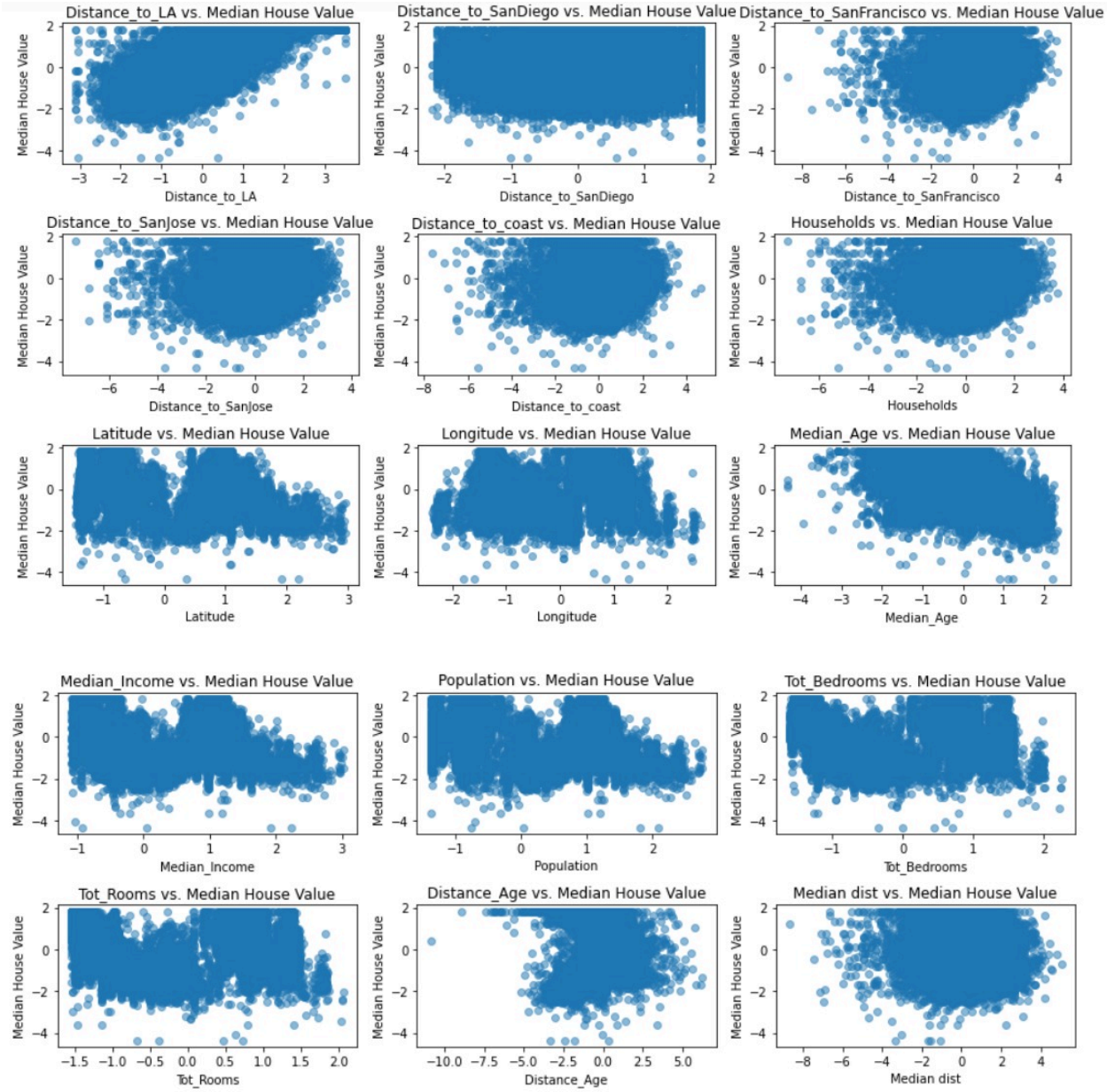
The relationships between each feature and the target variable (Median House Value) were visualized using scatter plots before and after data preprocessing. This allowed us to understand the relationships between features and the target variable and assess how preprocessing affected these relationships.

Before:





After:


Distance_to_LA vs. Median House Value
Median House Value
Distance_to_LA
Distance_to_SanDiego vs. Median House Value
Median House Value
Distance_to_SanDiego
Distance_to_SanFrancisco vs. Median House Value
Median House Value
Distance_to_SanFrancisco
Distance_to_SanJose vs. Median House Value
Median House Value
Distance_to_SanJose
Distance_to_coast vs. Median House Value
Median House Value
Distance_to_coast
Households vs. Median House Value
Median House Value
Households
Latitude vs. Median House Value
Median House Value
Latitude
Longitude vs. Median House Value
Median House Value
Longitude
Median_Age vs. Median House Value
Median House Value
Median_Age
Median_Income vs. Median House Value
Median House Value
Median_Income
Population vs. Median House Value
Median House Value
Population
Tot_Bedrooms vs. Median House Value
Median House Value
Tot_Bedrooms
Tot_Rooms vs. Median House Value
Median House Value
Tot_Rooms
Distance_Age vs. Median House Value
Median House Value
Distance_Age
Median dist vs. Median House Value
Median House Value
Median dist

Model Training:

1. Linear Regression:

It's a simple yet effective model, applied to the preprocessed data. The model was trained on the training set and evaluated on the validation set.
The validation results showed an **MSE** of approximately **0.1013** and an **MAE** of about **0.2339.**

2. Lasso Regression:

It's a regularization technique, employed with a grid search to find the best hyperparameter (alpha).
The best Lasso model exhibited an **MSE** of approximately **0.1013** and an **MAE** of about **0.2339** on the validation set.

3. Ridge Regression:

It's another regularization technique, also used with a grid search to determine the best hyperparameter (alpha).
The best Ridge model produced similar validation results, with an **MSE** of approximately **0.1013** and an **MAE** of about **0.2339**.

Performance Evaluation:

1. Validation Set Performance:

The validation set results for MSE and MAE were quite similar for all three models. The models did not significantly outperform each other on the validation set.

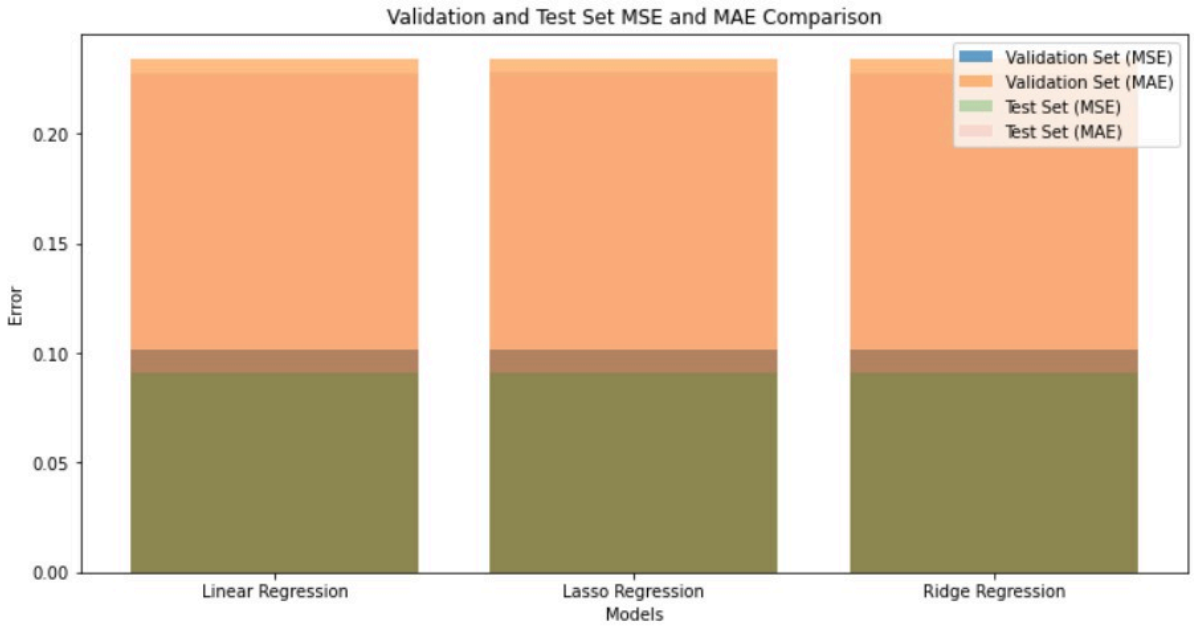
2. Validation Set Performance:

To assess the models' generalization to unseen data, the models were evaluated on the test set. Linear regression, lasso regression, and ridge regression produced similar test set results, with an MSE of approximately 0.0907 and an MAE of about 0.2276.

Residual Plot:

Residual plots were created to visualize the difference between actual and predicted values.

The plot shows that the models have consistent performance across the entire range of target values.



➢ Impact of Data Preprocessing:

It's important to note that the presented output, characterized by low Mean Squared Error (MSE) and Mean Absolute Error (MAE) values, was not our initial result.

Before conducting data preprocessing, the model's performance was considerably different. The following section illustrates the substantial impact of data preprocessing on model accuracy.

1. Initial Results:

Before data preprocessing, the models produced significantly higher MSE and MAE values, indicating poorer predictive performance. The MSE and MAE values in the initial results were substantially higher compared to the post-preprocessing results.

```
# Calculate mean squared error and mean absolute error for all models
linear_mse = mean_squared_error(y_val, linear_regressor_prediction)
linear_mae = mean_absolute_error(y_val, linear_regressor_prediction)
print("Mean Squared Error Of Linear Regression =", linear_mse)
print("Mean Absolute Error Of Linear Regression =", linear_mae)
```

```
Mean Squared Error Of Linear Regression = 4907211997.374585
Mean Absolute Error Of Linear Regression = 50790.06027105433
```

```
lasso_mse = mean_squared_error(y_val, lasso_regressor_prediction)
lasso_mae = mean_absolute_error(y_val, lasso_regressor_prediction)
print("Mean Squared Error Of Lasso Regression =", lasso_mse)
print("Mean Absolute Error Of Lasso Regression =", lasso_mae)
```

```
Mean Squared Error Of Lasso Regression = 4907219718.486601
Mean Absolute Error Of Lasso Regression = 50790.273473256915
```

```
ridge_mse = mean_squared_error(y_val, ridge_regressor_prediction)
ridge_mae = mean_absolute_error(y_val, ridge_regressor_prediction)
print("Mean Squared Error Of Ridge Regression =", ridge_mse)
print("Mean Absolute Error Of Ridge Regression =", ridge_mae)
```

```
Mean Squared Error Of Ridge Regression = 4907226928.247798
Mean Absolute Error Of Ridge Regression = 50790.607314504065
```

2. Post-Preprocessing Results:

Upon applying data preprocessing techniques such as log transformation to mitigate skewness, Z-score normalization to standardize features, and feature engineering to create new variables, the models exhibited a remarkable improvement in performance. The MSE and MAE values were considerably reduced, indicating that the models could make more accurate predictions.

```
linear_mse = mean_squared_error(y_val, linear_regressor_prediction)
linear_mae = mean_absolute_error(y_val, linear_regressor_prediction)
print("Mean Squared Error Of Linear Regression =", linear_mse)
print("Mean Absolute Error Of Linear Regression =", linear_mae)
```

```
Mean Squared Error Of Linear Regression = 0.10130599074050421
Mean Absolute Error Of Linear Regression = 0.23393468937926465
```

```
lasso_mse = mean_squared_error(y_val, lasso_regressor_prediction)
lasso_mae = mean_absolute_error(y_val, lasso_regressor_prediction)
print("Mean Squared Error Of Best Lasso Regression =", lasso_mse)
print("Mean Absolute Error Of Best Lasso Regression =", lasso_mae)
```

```
Lasso(alpha=0.0001, max_iter=10000)
Mean Squared Error Of Best Lasso Regression = 0.10127881291265355
Mean Absolute Error Of Best Lasso Regression = 0.23395521147021486
```

```
ridge_mse = mean_squared_error(y_val, ridge_regressor_prediction)
ridge_mae = mean_absolute_error(y_val, ridge_regressor_prediction)
print("Mean Squared Error Of Ridge Regression =", ridge_mse)
print("Mean Absolute Error Of Ridge Regression =", ridge_mae)
```

```
Mean Squared Error Of Ridge Regression = 0.10130438141390889
Mean Absolute Error Of Ridge Regression = 0.23393495832967653
```

In conclusion, the application of data preprocessing techniques was pivotal in achieving the low MSE and MAE values presented in this report. The preprocessing steps played a crucial role in enhancing the models' predictive power and making them suitable for practical use in predicting median house values in California.

Summary:

In this analysis, we applied linear regression, lasso regression, and ridge regression models to predict median house values in California.

All three models produced similar results on both the validation and test sets, with no significant differences in performance.

Further exploration and feature engineering may be required to enhance model performance.

The choice of the best model depends on the specific requirements and characteristics of the dataset.