Name : Cuadros Rivas, Alejandra Paola – KK5459
Homework - Feature Selection

Feature Selection

Objective

Understand and apply feature selection techniques (filter, embedded, or other methods) to optimize model performance and interpretability.

Instructions

1. ***Data Selection:***
   - *Choose a dataset from sources like UCI Machine Learning Repository, Kaggle, or any relevant open dataset of your interest.*
   - *Ensure the dataset has at least 10 features and a target variable.*
2. ***Feature Selection Methods:***
   - *Explore and implement at least two feature selection methods:*
     - ***Filter methods:*** *Use statistical measures like correlation, mutual information, or chi-square test.*
     - ***Wrapper methods****: recursive, forward, or backward selection.*
     - ***Embedded methods:*** *Use algorithms like decision trees, Lasso regression, or XGBoost for feature importance.*
3. ***Implementation Steps:***
   - *Preprocess the data (handle missing values, normalize/standardize if necessary).*
   - *Apply feature selection techniques and compare the results.*
   - *Train a machine learning model (e.g., logistic regression, decision tree, or a classifier of your choice) on the selected features.*
   - *Compare the model's performance before and after feature selection.*
4. ***Submission Requirements:***
   - *A detailed report with:*
     - *Dataset description.*
     - *Explanation of methods used for feature selection.*
     - *Results and performance comparison (accuracy, precision, recall, etc.).*
   - *Copy and paste the source code (Python) to your PDF or link from Google Collabs.*

Name : Cuadros Rivas, Alejandra Paola – KK5459
Homework - Feature Selection

Boston Housing Dataset

1. Introduction

Feature selection is one of the most significant processes in Machine Learning which concentrates on determining the best set of features in a given data set. Reducing the feature space allows for improving the performance, interpretability and the cost of the models.

This report investigates different feature selection methods that were implemented on the Boston Housing dataset. The goal is to estimate the median value of an owner-occupied home *(medv)* using a number of features that describe housing and neighbourhood characteristics.

2. Dataset Description

The Boston Housing dataset contains 506 observations and 14 variables (13 features and 1 target). The target variable is medv, representing the median home price in $1000s.

Features

- crim: Per capita crime rate by town.
- zn: Proportion of residential land zoned for lots > 25,000 sq. ft.
- indus: Proportion of non-retail business acres per town.
- chas: Charles River dummy variable (1 if bounds river; 0 otherwise).
- nox: Nitric oxide concentration (parts per 10 million).
- rm: Average number of rooms per dwelling.
- age: Proportion of owner-occupied units built before 1940.
- dis: Weighted distances to five Boston employment centres.
- rad: Accessibility to radial highways.
- tax: Property-tax rate per $10,000.
- ptratio: Pupil-teacher ratio by town.
- black: 1000(Bk - 0.63)^2, where Bk is the proportion of Black residents by town.
- lstat: % lower status of the population.

Target

- medv: Median value of owner-occupied homes in $1000s.

Name : Cuadros Rivas, Alejandra Paola – KK5459
Homework - Feature Selection

3. Methodology

Five feature selection techniques were applied. Each method was evaluated by training a Linear Regression model on the selected features.

Performance metrics for the regression model included:

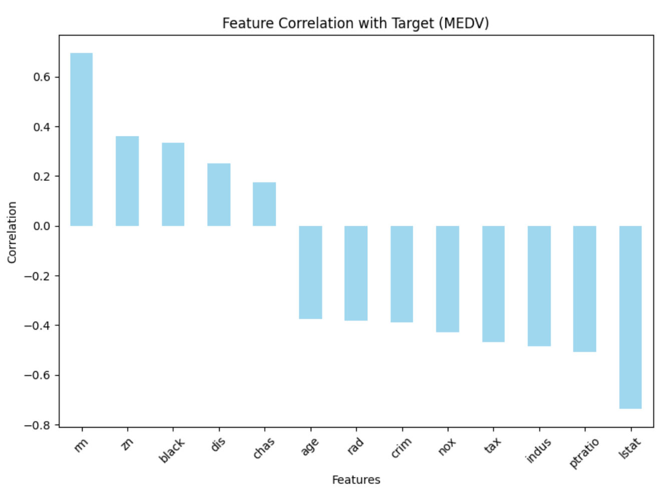
- RMSE (Root Mean Squared Error): Measures average prediction error.
- R^2 Score: Indicates how well the model explains variability in the target variable.

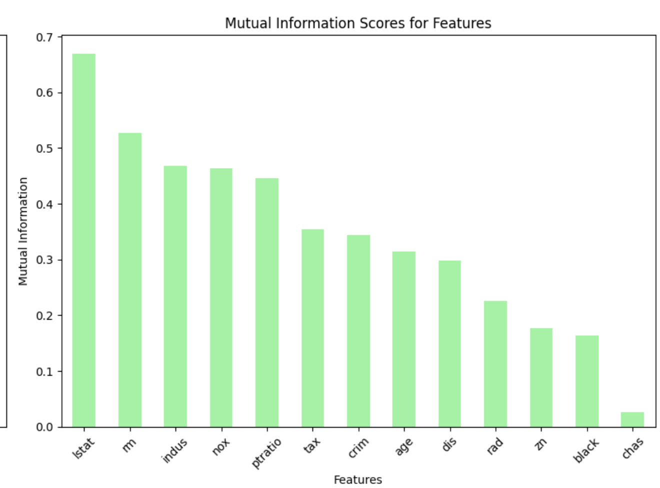
```python
# Train a Linear Regression model
baseline_model = LinearRegression()
baseline_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = baseline_model.predict(X_test)

# Evaluate the model
baseline_rmse = mean_squared_error(y_test, y_pred, squared=False)
baseline_r2 = r2_score(y_test, y_pred)

# Model Performance
print(f"RMSE: {baseline_rmse:.2f}")
print(f"R^2 Score: {baseline_r2:.2f}")
```

RMSE: 4.64

R^2 Score: 0.71

3.1 Filter Methods





Both correlation and mutual information identify rm and lstat as the most important features for predicting medv.

Correlation captures linear relationships, while mutual information highlights non-linear dependencies, revealing additional relevance for features like indus and nox.

Name : Cuadros Rivas, Alejandra Paola – KK5459
Homework - Feature Selection

3.1.1 **Correlation:** Features are ranked by their correlation with the target variable.

```python
# Select features
X_train_corr = pd.DataFrame(X_train, columns=X.columns)[top_corr_features]
X_test_corr = pd.DataFrame(X_test, columns=X.columns)[top_corr_features]

# Training
model_corr = LinearRegression()
model_corr.fit(X_train_corr, y_train)

# Predictions
y_pred_corr = model_corr.predict(X_test_corr)

# Model Performance
rmse_corr = mean_squared_error(y_test, y_pred_corr, squared=False)
r2_corr = r2_score(y_test, y_pred_corr)

print("\nModel Performance with Top Correlation Features:")
print(f"\n - RMSE: {rmse_corr:.2f}")
print(f"\n - R^2 Score: {r2_corr:.2f}")
```

Model Performance with Top Correlation Features:

- RMSE: 5.11
- R^2 Score: 0.65

3.1.2 **Mutual Information:** Measures non-linear dependencies between features and the target.

```python
# Select features
X_train_mi = pd.DataFrame(X_train, columns=X.columns)[top_mi_features]
X_test_mi = pd.DataFrame(X_test, columns=X.columns)[top_mi_features]

# Training
model_mi = LinearRegression()
model_mi.fit(X_train_mi, y_train)

# Predictions
y_pred_mi = model_mi.predict(X_test_mi)

# Model Performance
rmse_mi = mean_squared_error(y_test, y_pred_mi, squared=False)
r2_mi = r2_score(y_test, y_pred_mi)

print("Model Performance with Top Mutual Information Features:")
print(f"\n - RMSE: {rmse_mi:.2f}")
print(f"\n - R^2 Score: {r2_mi:.2f}")
```

Model Performance with Top Mutual Information Features:

- RMSE: 5.11
- R^2 Score: 0.65

Name : Cuadros Rivas, Alejandra Paola – KK5459
Homework - Feature Selection

3.2 Wrapper Method

Recursive Feature Elimination (RFE): Iteratively removes less important features using a model's coefficients.

```
# Linear Regression
estimator = LinearRegression()
selector = RFE(estimator, n_features_to_select=5, step=1)

# Fit RFE on training data
selector = selector.fit(X_train, y_train)

# Select features
rfe_selected_features = X.columns[selector.support_].tolist()
print("Features Selected by RFE:", rfe_selected_features)

# Training and evaluation
X_train_rfe = pd.DataFrame(X_train,
columns=X.columns)[rfe_selected_features]
X_test_rfe = pd.DataFrame(X_test,
columns=X.columns)[rfe_selected_features]

model_rfe = LinearRegression()
model_rfe.fit(X_train_rfe, y_train)
y_pred_rfe = model_rfe.predict(X_test_rfe)

# Model Performance
rmse_rfe = mean_squared_error(y_test, y_pred_rfe, squared=False)
r2_rfe = r2_score(y_test, y_pred_rfe)

print("\nModel Performance with RFE Features:")
print(f"\n - RMSE: {rmse_rfe:.2f}")
print(f"\n - R^2 Score: {r2_rfe:.2f}")
```

```
Features Selected by RFE:
 ['nox', 'rm', 'dis', 'ptratio', 'lstat']

Model Performance with RFE Features:
 - RMSE: 4.77
 - R^2 Score: 0.70
```

Name : Cuadros Rivas, Alejandra Paola – KK5459
Homework - Feature Selection

3.3 Embedded Methods

3.3.1 Lasso Regression (L1 Regularization): Shrinks less important feature coefficients to zero.

```
# Lasso Regression
lasso = Lasso(alpha=0.1, random_state=42)
lasso.fit(X_train, y_train)

# Feature Selection
lasso_selected_features = X.columns[lasso.coef_ != 0].tolist()
print("\nFeatures Selected by Lasso Regression:\n",
lasso_selected_features)

# Trainning and evaluation
X_train_lasso = pd.DataFrame(X_train,
columns=X.columns)[lasso_selected_features]
X_test_lasso = pd.DataFrame(X_test,
columns=X.columns)[lasso_selected_features]

model_lasso = LinearRegression()
model_lasso.fit(X_train_lasso, y_train)
y_pred_lasso = model_lasso.predict(X_test_lasso)

# Model Performance
rmse_lasso = mean_squared_error(y_test, y_pred_lasso, squared=False)
r2_lasso = r2_score(y_test, y_pred_lasso)

print("\nModel Performance with Lasso-Selected Features:")
print(f"\n - RMSE: {rmse_lasso:.2f}")
print(f"\n - R^2 Score: {r2_lasso:.2f}")
```

Features Selected by Lasso Regression:
['crim', 'zn', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax', 'ptratio', 'black', 'lstat']

Model Performance with Lasso-Selected Features:
- RMSE: 4.63
- R^2 Score: 0.71

Name : Cuadros Rivas, Alejandra Paola – KK5459
Homework - Feature Selection

3.3.2 **Random Forest Feature Importance:** Uses a tree-based model to rank features by their contribution to predictions.

```
# Random Forest Regressor
rf = RandomForestRegressor(random_state=42)
rf.fit(X_train, y_train)

# Feature importances
rf_feature_importances = pd.Series(rf.feature_importances_,
index=X.columns).sort_values(ascending=False)

# Select top 5 features
top_rf_features = rf_feature_importances.nlargest(5).index.tolist()
print("Top Features by Random Forest:", top_rf_features)

# Training and evaluation with selected features
X_train_rf = pd.DataFrame(X_train, columns=X.columns)[top_rf_features]
X_test_rf = pd.DataFrame(X_test, columns=X.columns)[top_rf_features]

model_rf = LinearRegression()
model_rf.fit(X_train_rf, y_train)
y_pred_rf = model_rf.predict(X_test_rf)

# Model Performance
rmse_rf = mean_squared_error(y_test, y_pred_rf, squared=False)
r2_rf = r2_score(y_test, y_pred_rf)

print("Model Performance with RF-Selected Features:")
print(f"RMSE: {rmse_rf:.2f}")
print(f"R^2 Score: {r2_rf:.2f}")
```

Top Features by Random Forest:
['rm', 'lstat', 'dis', 'crim', 'ptratio']

Model Performance with RF-Selected Features:
- RMSE: 5.05
- R^2 Score: 0.66

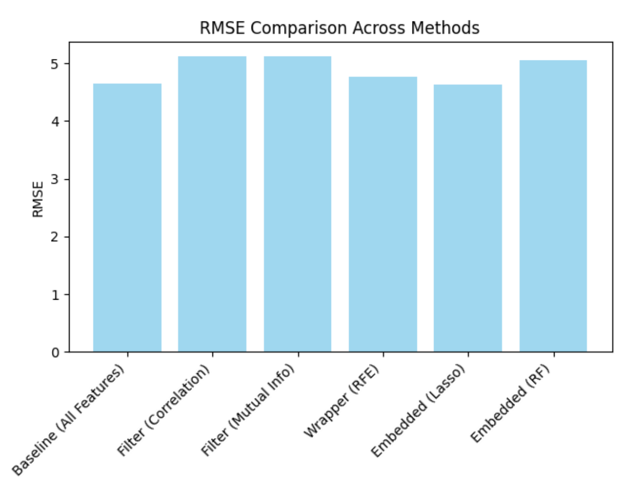
Name : Cuadros Rivas, Alejandra Paola – KK5459
Homework - Feature Selection

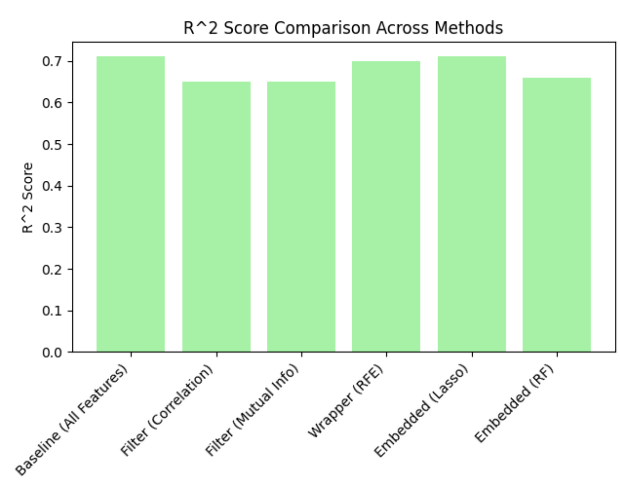
4. Results and Evaluation

Performance of Feature Selection Methods

Method	Selected Features	RMSE	R^2
Baseline (All Features)	All	4.64	0.71
Filter (Correlation)	*lstat, rm, ptratio, indus, tax*	5.11	0.65
Filter (Mutual Information)	*lstat, rm, indus, nox, ptratio*	5.11	0.65
Wrapper (RFE)	*nox, rm, dis, ptratio, lstat*	4.77	0.70
Embedded (Lasso Regression)	Most features except *indus*	4.63	0.71
Embedded (Random Forest Importance)	*rm, lstat, dis, crim, ptratio*	5.05	0.66

- With all features, the model achieved the best RMSE: 4.64 and R^2: 0.71.
- With Filter Methods, Correlation and Mutual Information selected overlapping features but underperformed RMSE: 5.11 and R^2: 0.65.
- With Wrapper Method the feature set was reduced to 5 features while maintaining strong performance RMSE: 4.77 and R^2: 0.70.
- With Embedded Methods, Lasso Regression matched the baseline model's performance while reducing the feature set, making it **the most effective technique** with RMSE: 4.63 and R^2: 0.71., while Random Forest identified relevant features but slightly underperformed RMSE: 5.05 and R^2: 0.66.





5. Conclusion

- Lasso Regression was the most effective feature selection method, achieving baseline performance with fewer features.

6. Appendix

Google Collab Link