Data Analysis report

Introduction

The objective of this analysis is to develop a prediction model for housing prices based on a data set which is Boston Housing. We will use different machine learning techniques, we try to understand the relationship between 13 housing features on the price of the house for ultimate price predictions for real estate transactions this was one of the data sets provided in which 506 observations and 13 features describing various aspects of the of this property in Boston. The model evaluated linear regression, ridge regression, polynomial regression

Methodology steps

In the preprocessing stage, we applied several techniques to clean and transform the raw data set in preparation for modeling our ultimate goal is data is suitable for training machine learning.

In the very beginning, i prepared a data model with linear regression and performed various evaluation parameters after performing the evaluation the result is shown below

To perform linear regression and its evaluation steps are as follows

1. At the very first step I load the data from the URL which was mentioned using pd.read_csv()

```
data_url = "http://lib.stat.cmu.edu/datasets/boston"

# Read the dataset directly from the URL and preprocess it
raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
```

2. Preprocess the raw data all those 13 columns holding different features with this i extract the data feature and target feature

```
# Extract the necessary data and target variables from the raw DataFrame
data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
target = raw_df.values[1::2, 2]

# Define column names for the features
```

3. Now after loading next step is prepare data coloums need to mentioned for feature and target

```
# Define column names for the features
feature_names = [
    'CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT'
]
arget_name = 'MEDV'
```

4. Next step data need to be split in two parts train and test in my case 80 % train data 20 % test for which data need to be split using train_test_spilt() method and (test _size=0.2).

```
# Split the data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

5. Then i create an instance of LinearRegression model

```
# Create a LinearRegression model
model = LinearRegression()
```

6. Train model fit the Linear_regression model on a trainng data (x_train, y_train)using model.fit()

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

7. Make predictions y_pred on the testing data x_test with model.predict()

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

8. Then basic evalution have been performed such as R_SQUARED , Mean Absolute Error, Root Mean Squared Error

```
# Evaluate the model
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
mape_linear = np.mean(np.abs((y_test - y_pred) / y_test)) * 100
rmse_linear = np.sqrt(mean_squared_error(y_test, y_pred))
print("Linear Regression")

print(f"RMSE: {rmse_linear:.4f}")
print(f"MAPE: {mape_linear:.4f}%")
# Print evaluation metrics
print(f"R-squared: {r2:.4f}")
print(f"Mean Squared Error: {mse:.4f}")
print(f"Mean Absolute Error: {mae:.4f}")
print(f"RMSE: {rmse_linear:.4f}")
print(f"MAPE: {mape_linear:.4f}%")
```

9. Result are shared below

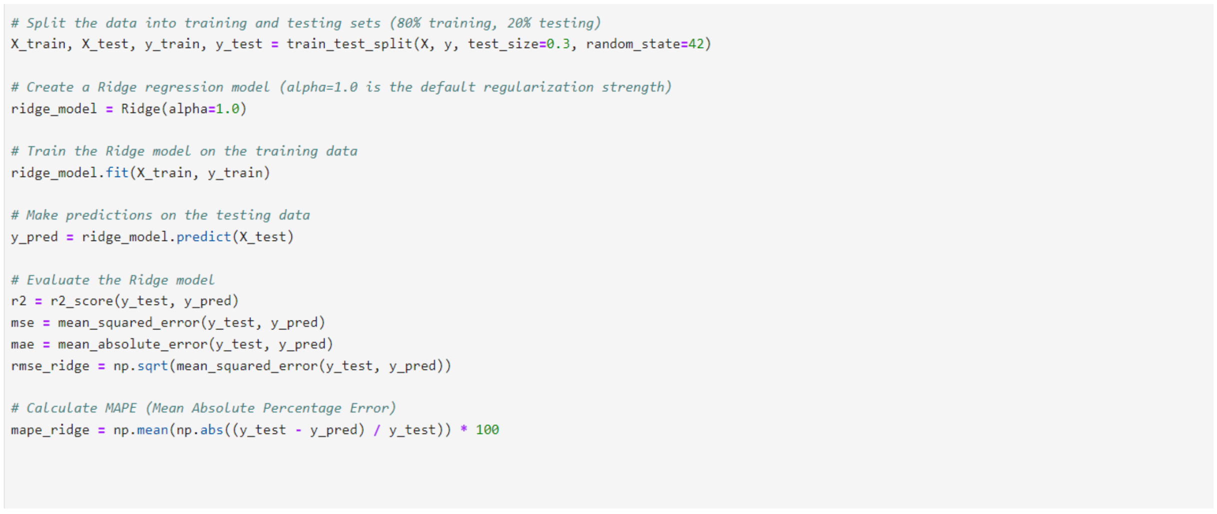
```
Linear Regression
RMSE: 4.9286
MAPE: 16.8664%
R-squared: 0.6688
Mean Squared Error: 24.2911
Mean Absolute Error: 3.1891
RMSE: 4.9286
MAPE: 16.8664%
```

10. In addition to this linear regression with grid search is also perform so can we auto tune our model and find the best model using grid search.

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Linear Regression with Grid Search
RMSE: 4.9286
R-squared: 0.6688
Mean Squared Error: 24.2911
Mean Absolute Error: 3.1891
DataFrame saved as PDF: boston_housing_data_with_gridsearch.pdf
```

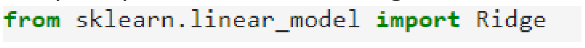
The second model we prepare usind the ridge regression and performed grid search also both results are shared below those steps which are common in both linear regression and ridge have not mentioned only the difference in both the model has been written below.

1. The steps which are similar such as loading dataset and spilting the data into train and test is same as we have in linear regression result are below at last we try to evaluate which datamodel is the best.



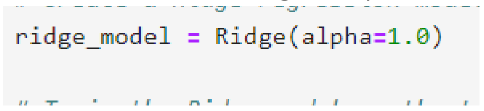
```
# Split the data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Create a Ridge regression model (alpha=1.0 is the default regularization strength)
ridge_model = Ridge(alpha=1.0)

# Train the Ridge model on the training data
ridge_model.fit(X_train, y_train)

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

# Evaluate the Ridge model
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
rmse_ridge = np.sqrt(mean_squared_error(y_test, y_pred))

# Calculate MAPE (Mean Absolute Percentage Error)
mape_ridge = np.mean(np.abs((y_test - y_pred) / y_test)) * 100
```

2. We will initialize ridge regression which contains L2 regulations to control model complexity and reduce overfitting.



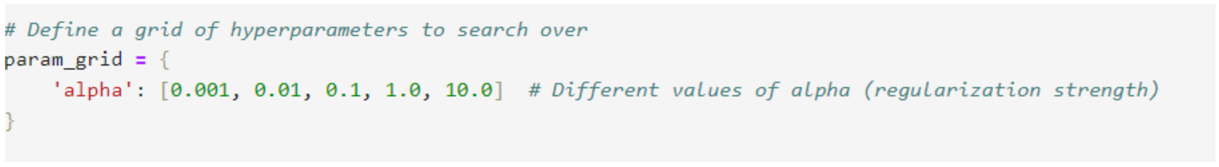
```
from sklearn.linear_model import Ridge
```

3. I define the various range of alpha values and result are used.



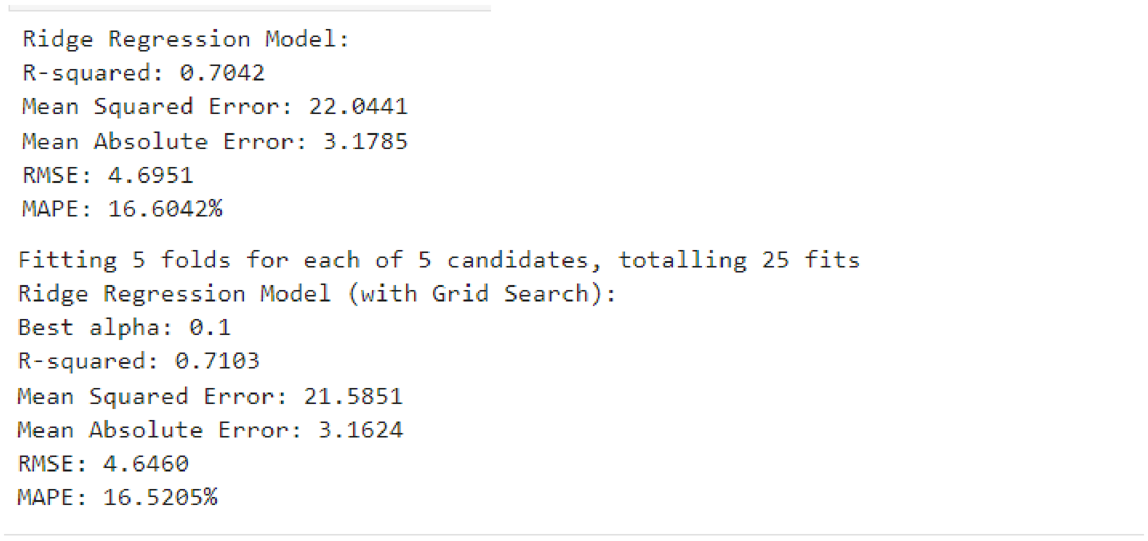
```
ridge_model = Ridge(alpha=1.0)
```

4. With this later on grid search is also performed result are shared for the analysis and i try to minimizes the mean squared error on the validaton set with values.



```
# Define a grid of hyperparameters to search over
param_grid = {
    'alpha': [0.001, 0.01, 0.1, 1.0, 10.0]  # Different values of alpha (regularization strength)
}
```

5. I try to otimizes the alpha value to reach the best data model.



```
Ridge Regression Model:
R-squared: 0.7042
Mean Squared Error: 22.0441
Mean Absolute Error: 3.1785
RMSE: 4.6951
MAPE: 16.6042%

Fitting 5 folds for each of 5 candidates, totalling 25 fits
Ridge Regression Model (with Grid Search):
Best alpha: 0.1
R-squared: 0.7103
Mean Squared Error: 21.5851
Mean Absolute Error: 3.1624
RMSE: 4.6460
MAPE: 16.5205%
```

The third data model we use using the polynomial regression using linear regression pipeline.

1. Same as linear regression load and preprocess the dataset to extract features (x) and target variable (y).

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error

# Assuming boston_df is your DataFrame containing the Boston housing dataset
# Define features (X) and target variable (y)
X = boston_df.drop(columns=['MEDV'])  # Features (all columns except 'MEDV')
y = boston_df['MEDV']  # Target variable ('MEDV')
```

2. Then i performed spilit of the data into training and testing sets for model evalution.

```
# Split the data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

3. Then i apply polynomial feature to generate high degree polynomial feature from original dataset to capture non linear relationship between variables.

```
# Create a Polynomial Regression pipeline
# The pipeline includes PolynomialFeatures and LinearRegressio
degree = 2  # Degree of the polynomial features
polynomial_regression = Pipeline([
    ('poly_features', PolynomialFeatures(degree=degree)),
    ('linear_regression', LinearRegression())
])
```

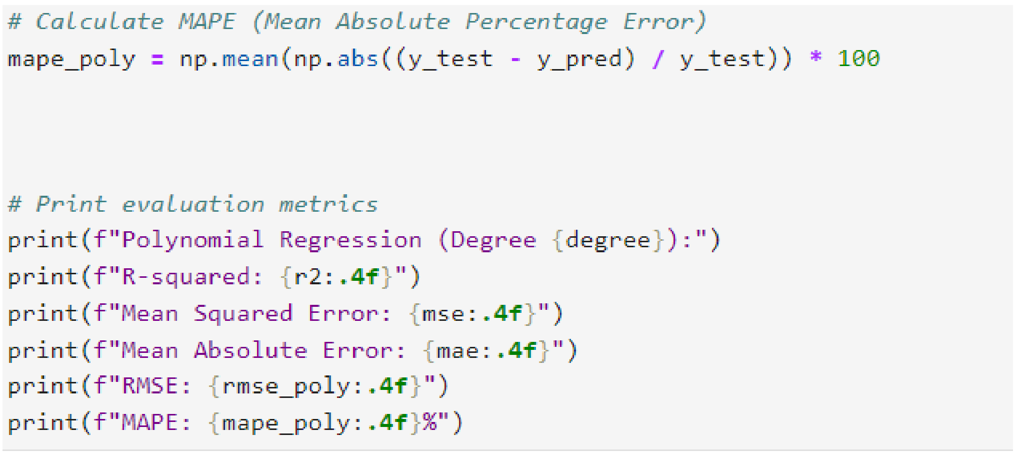
4. Then i will create a pipeline that combines polynomial features with linear regression to form a polynomial regression model.
5. With this we provide training to data model using x train and y train

```
# Train the Polynomial Regression pipeline on the training data
polynomial_regression.fit(X_train, y_train)
```

6. After training the data we will use this Polynomial Regression Model to make predictions on the testing data (x_test)

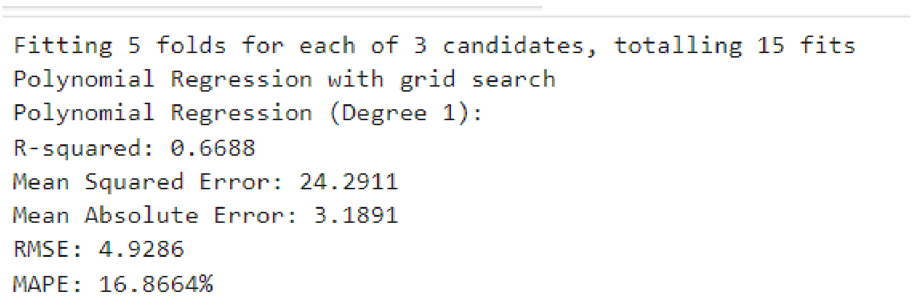
```
# Make predictions on the testing data
y_pred = polynomial_regression.predict(X_test)
```

7. We will evalute the polynomial regression model using the same metrics as linear regression such as R-squared, MSE , MAE, RMSE



```
# Calculate MAPE (Mean Absolute Percentage Error)
mape_poly = np.mean(np.abs((y_test - y_pred) / y_test)) * 100



# Print evaluation metrics
print(f"Polynomial Regression (Degree {degree}):")
print(f"R-squared: {r2:.4f}")
print(f"Mean Squared Error: {mse:.4f}")
print(f"Mean Absolute Error: {mae:.4f}")
print(f"RMSE: {rmse_poly:.4f}")
print(f"MAPE: {mape_poly:.4f}%")
```

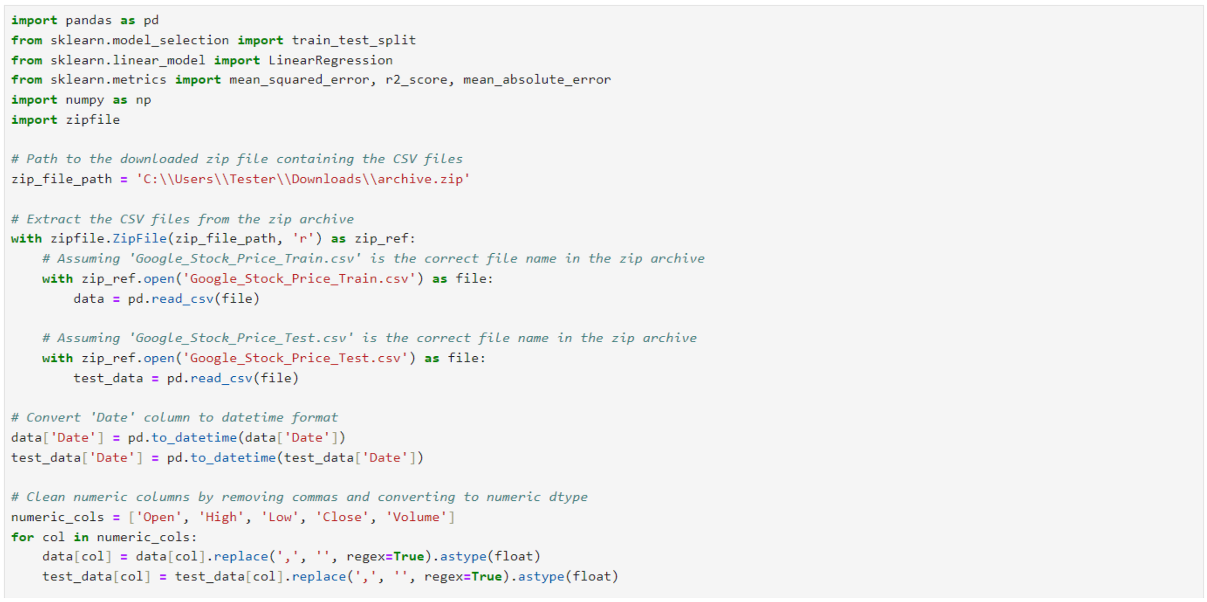
```
Polynomial Regression (Degree 2):
R-squared: 0.8066
Mean Squared Error: 14.1836
Mean Absolute Error: 2.5879
RMSE: 3.7661
MAPE: 13.8444%
```



```
Fitting 5 folds for each of 3 candidates, totalling 15 fits
Polynomial Regression with grid search
Polynomial Regression (Degree 1):
R-squared: 0.6688
Mean Squared Error: 24.2911
Mean Absolute Error: 3.1891
RMSE: 4.9286
MAPE: 16.8664%
```

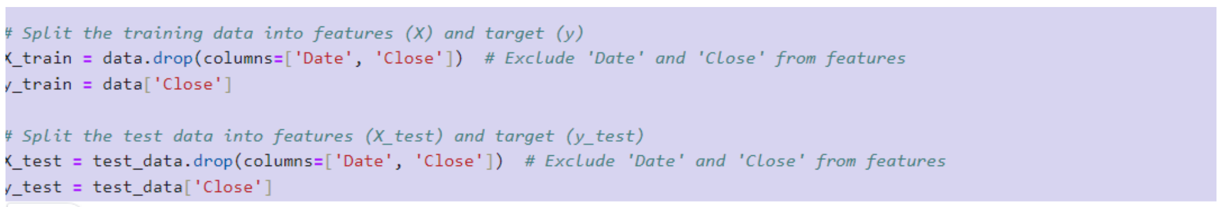
<u>Steps performed on other data set google _stock_flow</u>

1. For this data set we have to download this data set which is in zip folder contain two data set inside test and train first we will build a model based on linear regression.



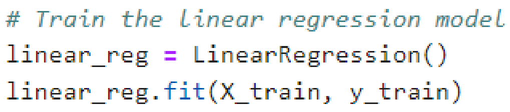
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
import numpy as np
import zipfile

# Path to the downloaded zip file containing the CSV files
zip_file_path = 'C:\\Users\\Tester\\Downloads\\archive.zip'

# Extract the CSV files from the zip archive
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    # Assuming 'Google_Stock_Price_Train.csv' is the correct file name in the zip archive
    with zip_ref.open('Google_Stock_Price_Train.csv') as file:
        data = pd.read_csv(file)

    # Assuming 'Google_Stock_Price_Test.csv' is the correct file name in the zip archive
    with zip_ref.open('Google_Stock_Price_Test.csv') as file:
        test_data = pd.read_csv(file)

# Convert 'Date' column to datetime format
data['Date'] = pd.to_datetime(data['Date'])
test_data['Date'] = pd.to_datetime(test_data['Date'])

# Clean numeric columns by removing commas and converting to numeric dtype
numeric_cols = ['Open', 'High', 'Low', 'Close', 'Volume']
for col in numeric_cols:
    data[col] = data[col].replace(',', '', regex=True).astype(float)
    test_data[col] = test_data[col].replace(',', '', regex=True).astype(float)
```

2. Then we will spilt the data into target and feature to train our model.



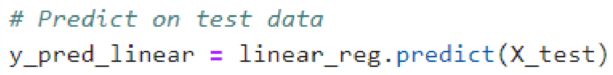
```
# Split the training data into features (X) and target (y)
X_train = data.drop(columns=['Date', 'Close'])  # Exclude 'Date' and 'Close' from features
y_train = data['Close']

# Split the test data into features (X_test) and target (y_test)
X_test = test_data.drop(columns=['Date', 'Close'])  # Exclude 'Date' and 'Close' from features
y_test = test_data['Close']
```

3. Then we will train our data model.



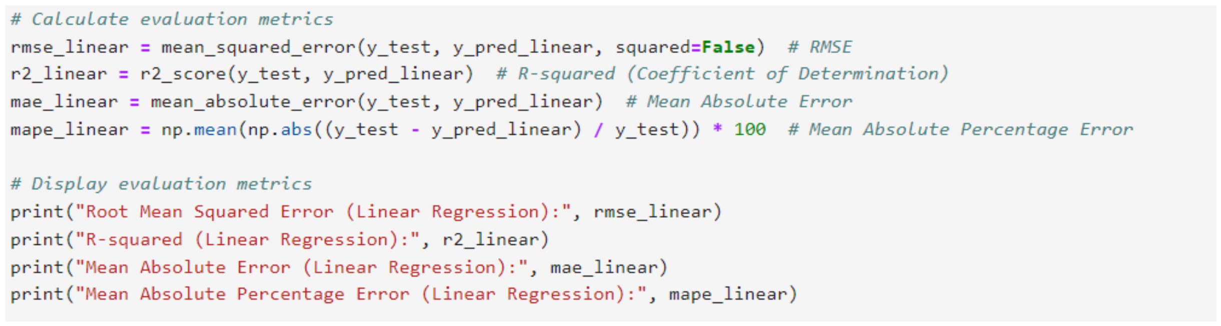
```
# Train the linear regression model
linear_reg = LinearRegression()
linear_reg.fit(X_train, y_train)
```

4. Then we will do predictions out of that data model.

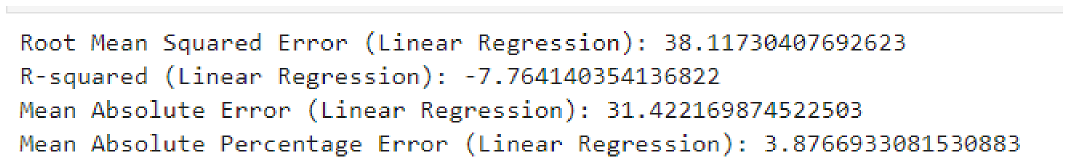


```
# Predict on test data
y_pred_linear = linear_reg.predict(X_test)
```

5. Then we will evaluate for best data model.



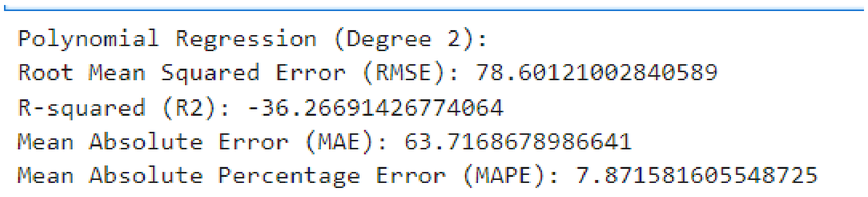
```
# Calculate evaluation metrics
rmse_linear = mean_squared_error(y_test, y_pred_linear, squared=False)  # RMSE
r2_linear = r2_score(y_test, y_pred_linear)  # R-squared (Coefficient of Determination)
mae_linear = mean_absolute_error(y_test, y_pred_linear)  # Mean Absolute Error
mape_linear = np.mean(np.abs((y_test - y_pred_linear) / y_test)) * 100  # Mean Absolute Percentage Error

# Display evaluation metrics
print("Root Mean Squared Error (Linear Regression):", rmse_linear)
print("R-squared (Linear Regression):", r2_linear)
print("Mean Absolute Error (Linear Regression):", mae_linear)
print("Mean Absolute Percentage Error (Linear Regression):", mape_linear)
```



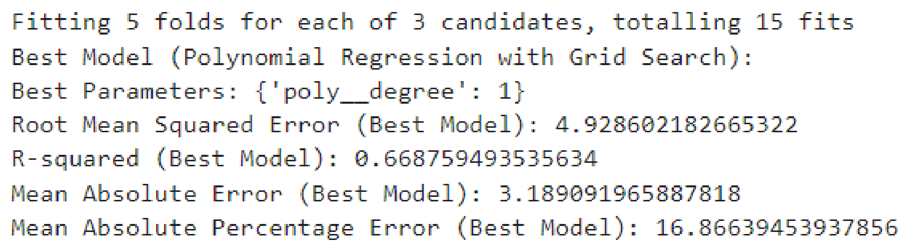
```
Root Mean Squared Error (Linear Regression): 38.11730407692623
R-squared (Linear Regression): -7.764140354136822
Mean Absolute Error (Linear Regression): 31.422169874522503
Mean Absolute Percentage Error (Linear Regression): 3.8766933081530883
```

```
Root Mean Squared Error (Linear Regression): 38.11730407692623
R-squared (Linear Regression): -7.764140354136822
Mean Absolute Error (Linear Regression): 31.422169874522503
Mean Absolute Percentage Error (Linear Regression): 3.8766933081530883
```

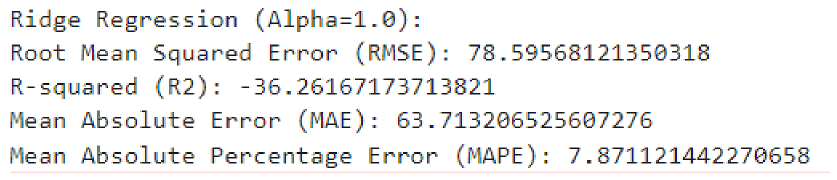
I performed the same steps for this data model polynomial regression and ridge regression first without grid search then with grid search for analysis of data in proper way here is the following result with both the regression.



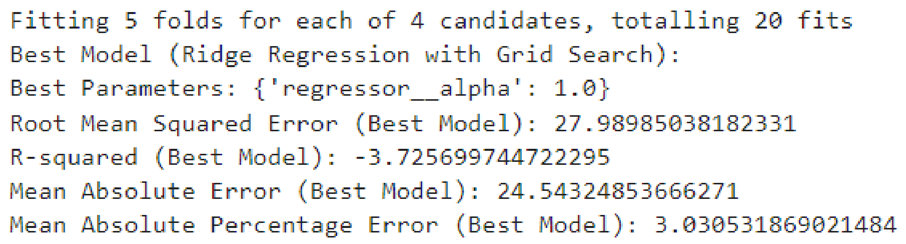
```
Polynomial Regression (Degree 2):
Root Mean Squared Error (RMSE): 78.60121002840589
R-squared (R2): -36.26691426774064
Mean Absolute Error (MAE): 63.7168678986641
Mean Absolute Percentage Error (MAPE): 7.871581605548725
```



```
Fitting 5 folds for each of 3 candidates, totalling 15 fits
Best Model (Polynomial Regression with Grid Search):
Best Parameters: {'poly__degree': 1}
Root Mean Squared Error (Best Model): 4.928602182665322
R-squared (Best Model): 0.668759493535634
Mean Absolute Error (Best Model): 3.189091965887818
Mean Absolute Percentage Error (Best Model): 16.86639453937856
```



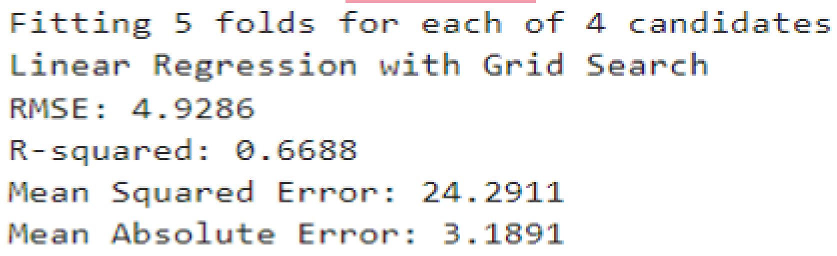
```
Ridge Regression (Alpha=1.0):
Root Mean Squared Error (RMSE): 78.59568121350318
R-squared (R2): -36.26167173713821
Mean Absolute Error (MAE): 63.713206525607276
Mean Absolute Percentage Error (MAPE): 7.871121442270658
```



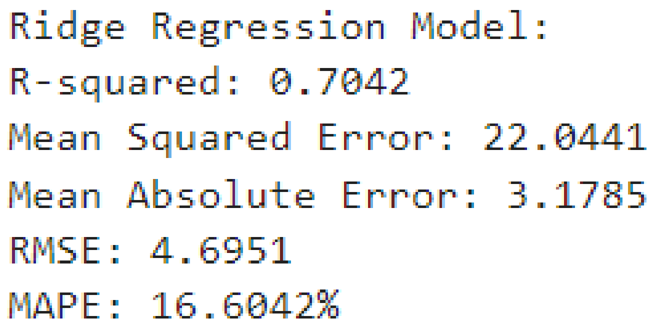
```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Best Model (Ridge Regression with Grid Search):
Best Parameters: {'regressor__alpha': 1.0}
Root Mean Squared Error (Best Model): 27.98985038182331
R-squared (Best Model): -3.725699744722295
Mean Absolute Error (Best Model): 24.54324853666271
Mean Absolute Percentage Error (Best Model): 3.030531869021484
```

Based on the evaluation made on the first data set to find which data model is best with the reason why?

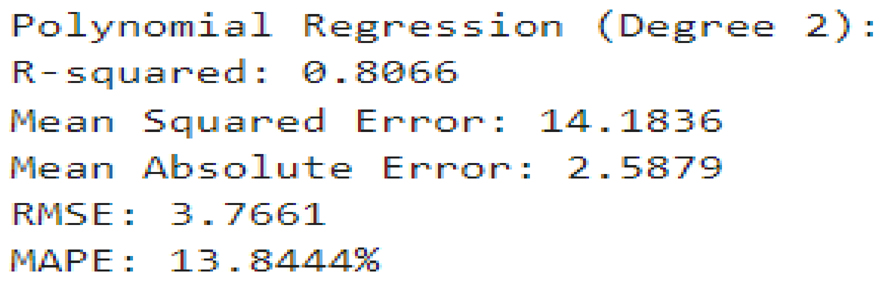
1. **Boston housing evaluation data analysis**



```
Fitting 5 folds for each of 4 candidates
Linear Regression with Grid Search
RMSE: 4.9286
R-squared: 0.6688
Mean Squared Error: 24.2911
Mean Absolute Error: 3.1891
```



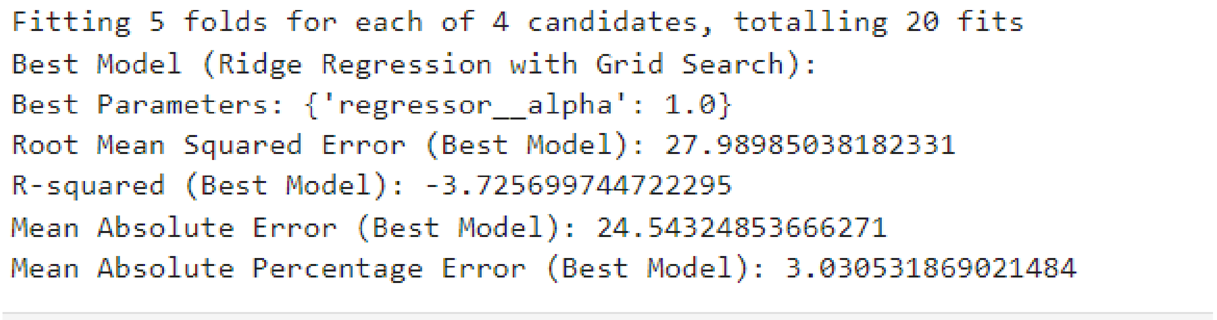
```
Ridge Regression Model:
R-squared: 0.7042
Mean Squared Error: 22.0441
Mean Absolute Error: 3.1785
RMSE: 4.6951
MAPE: 16.6042%
```



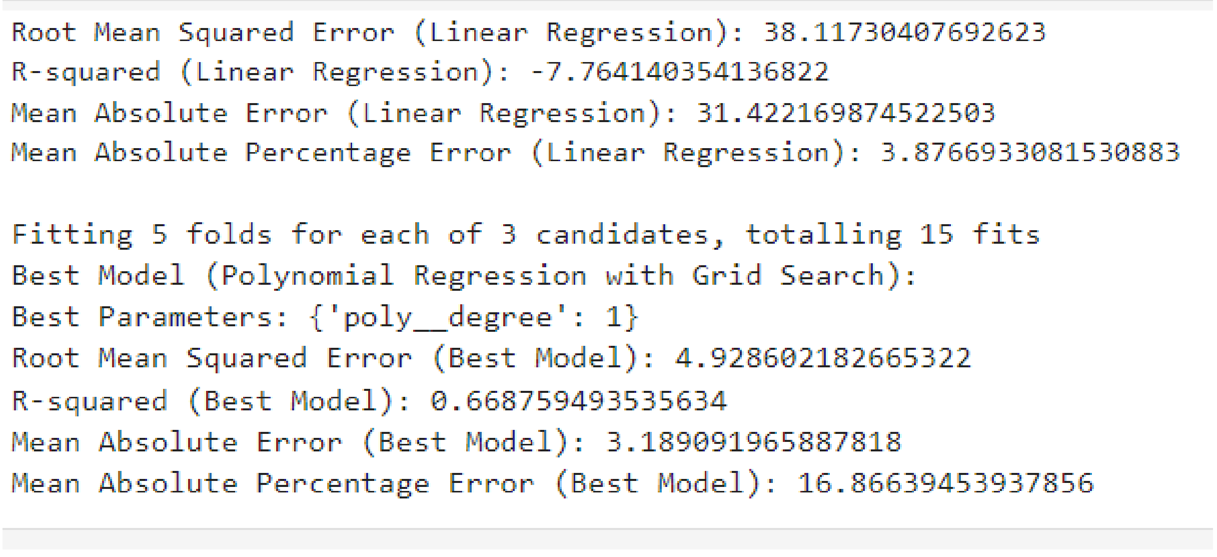
```
Polynomial Regression (Degree 2):
R-squared: 0.8066
Mean Squared Error: 14.1836
Mean Absolute Error: 2.5879
RMSE: 3.7661
MAPE: 13.8444%
```

based on evaluation metrics which indicates the proportion of the variance in the dependent variable which we predict from the independent variables. The higher R-squared values suggest a better fit of the model to the data. In our case, the polynomial regression with the degree 2 model has the highest R squared value of 0.8066, including the best overall fit among the models.
RMSE which stands for root mean squared error it is another parameter lower RMSE values indicate better model performance so here also polynomial regression degree 2 model has the lowest RMSE 3.7661 it suggests that it will give more accurate results as compared to other models.
MAPE(Mean Absolute Percentage Error):- this metric measures the accuracy of predictions as a percentage and lower values indicate better accuracy here in our case lower is 13.844%, which indicates superior accuracy in predicting target variables.

2. **<u>Result analysis for 2nd data set google_stock_flow for best data model and why?</u>**



```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Best Model (Ridge Regression with Grid Search):
Best Parameters: {'regressor__alpha': 1.0}
Root Mean Squared Error (Best Model): 27.98985038182331
R-squared (Best Model): -3.725699744722295
Mean Absolute Error (Best Model): 24.54324853666271
Mean Absolute Percentage Error (Best Model): 3.030531869021484
```

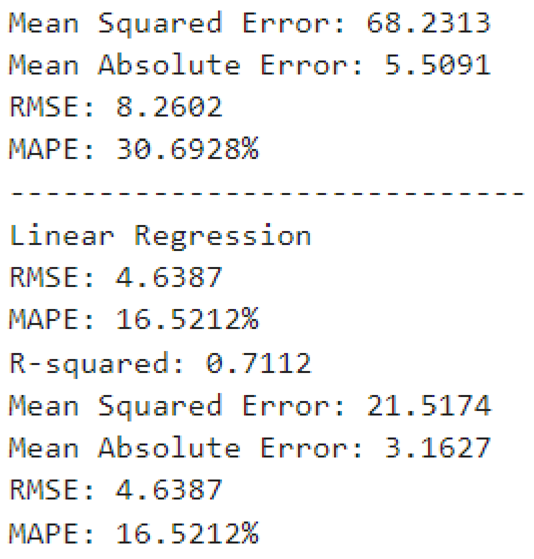


```
Root Mean Squared Error (Linear Regression): 38.11730407692623
R-squared (Linear Regression): -7.764140354136822
Mean Absolute Error (Linear Regression): 31.422169874522503
Mean Absolute Percentage Error (Linear Regression): 3.8766933081530883

Fitting 5 folds for each of 3 candidates, totalling 15 fits
Best Model (Polynomial Regression with Grid Search):
Best Parameters: {'poly__degree': 1}
Root Mean Squared Error (Best Model): 4.928602182665322
R-squared (Best Model): 0.668759493535634
Mean Absolute Error (Best Model): 3.189091965887818
Mean Absolute Percentage Error (Best Model): 16.86639453937856
```

Root mean Squared Error(RMSE): lower value indicated better model performance in terms of prediction accuracy. The “Best Model” (Ridge regression with grid search significantly lower RMSE of 27.9898 compared to others
Mean absolute Error(MAE) and Mean Absolute Percentage ERROR(MAPE). The “best model” also exhibits lower which signalises the better model in our case Ridge Regression is better.

3. **<u>Result as we increase or decrease the volume of data set we will have the following data result.</u>**

```
Training Data Size: 30.0%
Polynomial Regression (Degree 2):
R-squared: -5.8127
Mean Squared Error: 584.9040
Mean Absolute Error: 12.7486
RMSE: 24.1848
MAPE: 81.4619%
-----------------------------
Training Data Size: 40.0%
Polynomial Regression (Degree 2):
R-squared: -0.2162
Mean Squared Error: 99.2433
Mean Absolute Error: 5.4619
RMSE: 9.9621
MAPE: 33.8957%
-----------------------------
Training Data Size: 50.0%
Polynomial Regression (Degree 2):
R-squared: -0.9447
Mean Squared Error: 157.7765
Mean Absolute Error: 7.3773
RMSE: 12.5609
MAPE: 43.7372%
-----------------------------
Training Data Size: 60.0%
Polynomial Regression (Degree 2):
R-squared: -0.0091
Mean Squared Error: 76.6342
Mean Absolute Error: 5.6376
RMSE: 8.7541
MAPE: 35.8019%
-----------------------------
Training Data Size: 70.0%
Polynomial Regression (Degree 2):
R-squared: 0.0843
Mean Squared Error: 68.2313
```



```
Mean Squared Error: 68.2313
Mean Absolute Error: 5.5091
RMSE: 8.2602
MAPE: 30.6928%
-----------------------------
Linear Regression
RMSE: 4.6387
MAPE: 16.5212%
R-squared: 0.7112
Mean Squared Error: 21.5174
Mean Absolute Error: 3.1627
RMSE: 4.6387
MAPE: 16.5212%
```

Analysis from the above data on the best model that was selected.

As we increase the data size the performance of the data model deteriorates, with a negative value in R-squared values indicating poor fit.

When we increase the data size it is not consistent

High RMSE and MAPE percentages suggest that the model predictions are significantly off from actual value.

4. In the same way we found that the best model for data set 2 is ridge regression we analyze the result from increasing and decreasing the data volume and its impact.

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Training Data Size: 20.0%
Best Model (Ridge Regression with Grid Search):
Best Parameters: {'regressor__alpha': 0.1}
Root Mean Squared Error (Best Model): 41.458002741319426
R-squared (Best Model): -9.367683300766823
Mean Absolute Error (Best Model): 38.86178155343731
Mean Absolute Percentage Error (Best Model): 4.796311900467221
-----------------------------
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Training Data Size: 30.0%
Best Model (Ridge Regression with Grid Search):
Best Parameters: {'regressor__alpha': 0.01}
Root Mean Squared Error (Best Model): 33.41078823242439
R-squared (Best Model): -5.733462006995758
Mean Absolute Error (Best Model): 23.56427651108084
Mean Absolute Percentage Error (Best Model): 2.9024533305873086
-----------------------------
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Training Data Size: 40.0%
Best Model (Ridge Regression with Grid Search):
Best Parameters: {'regressor__alpha': 0.01}
Root Mean Squared Error (Best Model): 33.89314431079394
R-squared (Best Model): -5.929289275787051
Mean Absolute Error (Best Model): 24.152317327213712
Mean Absolute Percentage Error (Best Model): 2.975330372960038
-----------------------------
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Training Data Size: 50.0%
Best Model (Ridge Regression with Grid Search):
Best Parameters: {'regressor__alpha': 0.01}
Root Mean Squared Error (Best Model): 34.79861419158879
R-squared (Best Model): -6.304472674863422
Mean Absolute Error (Best Model): 27.284810132598047
Mean Absolute Percentage Error (Best Model): 3.3641809875295343
```

```
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Training Data Size: 50.0%
Best Model (Ridge Regression with Grid Search):
Best Parameters: {'regressor__alpha': 0.01}
Root Mean Squared Error (Best Model): 34.79861419158879
R-squared (Best Model): -6.304472674863422
Mean Absolute Error (Best Model): 27.284810132598047
Mean Absolute Percentage Error (Best Model): 3.3641809875295343
-----------------------------
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Training Data Size: 60.0%
Best Model (Ridge Regression with Grid Search):
Best Parameters: {'regressor__alpha': 0.01}
Root Mean Squared Error (Best Model): 35.8010130049876
R-squared (Best Model): -6.731354753101275
Mean Absolute Error (Best Model): 29.011950069461875
Mean Absolute Percentage Error (Best Model): 3.5783104530371306
-----------------------------
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Training Data Size: 70.0%
Best Model (Ridge Regression with Grid Search):
Best Parameters: {'regressor__alpha': 0.01}
Root Mean Squared Error (Best Model): 34.05477604999854
R-squared (Best Model): -5.995536515240621
Mean Absolute Error (Best Model): 25.71744500303678
Mean Absolute Percentage Error (Best Model): 3.1698718066538443
-----------------------------
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Training Data Size: 80.0%
Best Model (Ridge Regression with Grid Search):
Best Parameters: {'regressor__alpha': 0.01}
Root Mean Squared Error (Best Model): 33.15231313735974
R-squared (Best Model): -5.629681157059344
Mean Absolute Error (Best Model): 25.697932299050997
Mean Absolute Percentage Error (Best Model): 3.1678483272583957
-----------------------------
```

Training Data Size	RMSE	R-squared	MAE	MAPE (%)
20%	41.46	-9.37	38.86	4.80
30%	33.41	-5.73	23.56	2.90
40%	33.89	-5.93	24.15	2.98
50%	34.80	-6.30	27.28	3.36
60%	35.80	-6.73	29.01	3.58
70%	34.05	-6.00	25.72	3.17
80%	33.15	-5.63	25.70	3.17

Summary table for better analysis.
Model performance improves with larger training data sizes, as seen in decreasing RMSE and increasing R-squared values.
The best model 30 %training data achieved an RMSE of 33.41 and an R-squared of -5.73, indicating moderate predictive accuracy.

Hyperparameter tuning optimal alpha =0.01 significantly influenced model performance. Overall we analyzed that there is an impact of the data volume is there on the data set and also there are different models to calculate best.