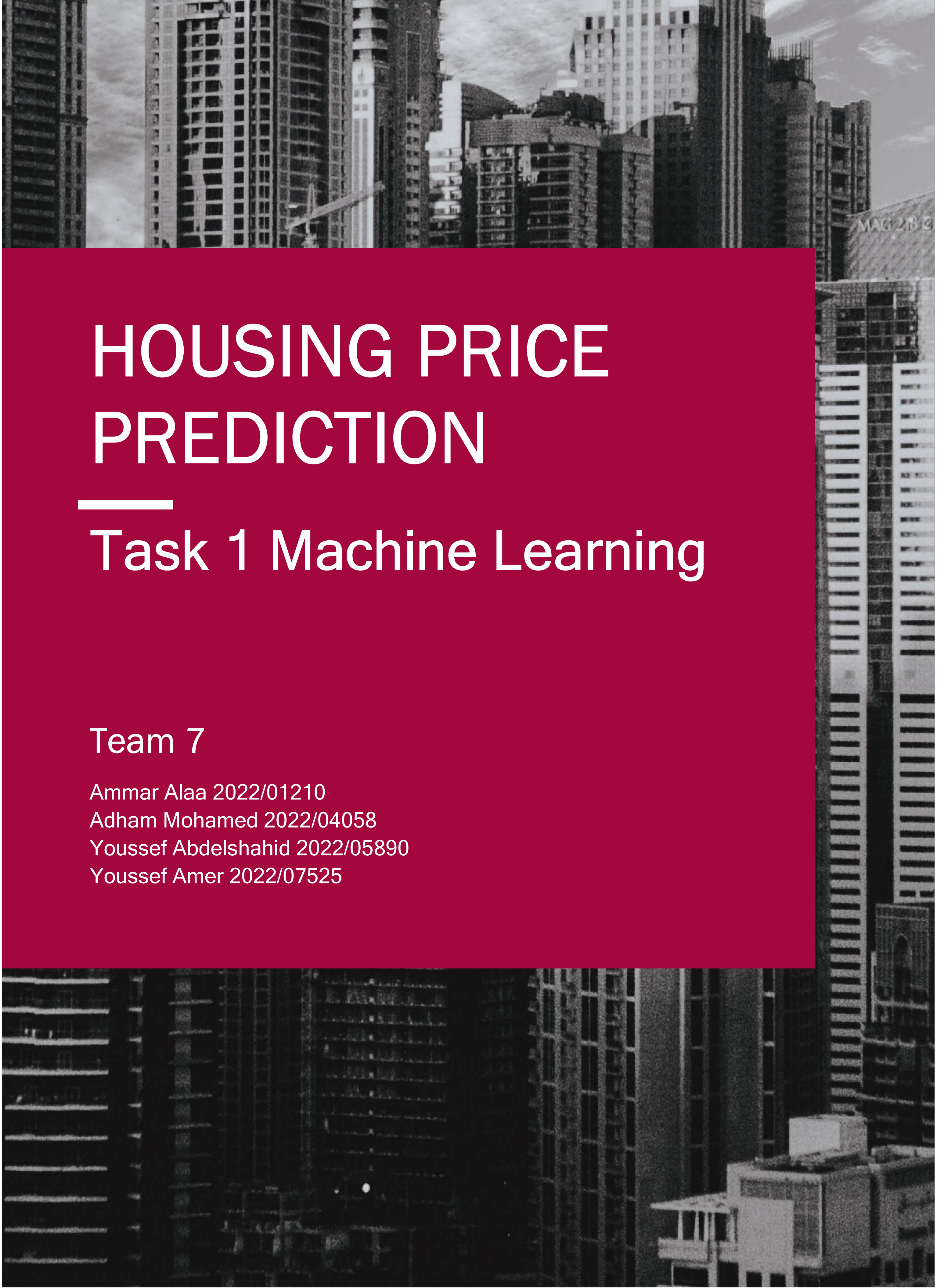


HOUSING PRICE PREDICTION

Task 1 Machine Learning

Team 7

Ammar Alaa 2022/01210
Adham Mohamed 2022/04058
Youssef Abdelshahid 2022/05890
Youssef Amer 2022/07525

Introduction

Objective

To predict housing prices based on house features and specifications (250k as an example) using classification techniques.

OVERVIEW

Our datasets involve the data of many houses and their features like the year they were build, neighbourhood, area size, kitchen quality, etc.

Data Exploration and Statistical Analysis

DATASET DESCRIPTION

The dataset consists of 81 columns and 1460 rows

Data columns (total 81 columns):

Id MSSubClass MSZoning LotFrontage LotArea Street
Alley LotShape LandContour Utilities LotConfig LandSlope
Neighborhood Condition1 Condition2 BldgType HouseStyle
OverallQual OverallCond YearBuilt YearRemodAdd
RoofStyle RoofMatl Exterior1st Exterior2nd MasVnrType
MasVnrArea ExterQual ExterCond Foundation BsmtQual
BsmtCond BsmtExposure BsmtFinType1 BsmtFinSF1
BsmtFinType2 BsmtFinSF2 BsmtUnfSF TotalBsmtSF
Heating HeatingQC CentralAir Electrical 1stFlrSF 2ndFlrSF
LowQualFinSF GrLivArea BsmtFullBath BsmtHalfBath
FullBath HalfBath BedroomAbvGr KitchenAbvGr KitchenQual
TotRmsAbvGrd Functional Fireplaces FireplaceQu GarageType
GarageYrBlt GarageFinish GarageCars GarageArea
GarageQual GarageCond PavedDrive WoodDeckSF
OpenPorchSF EnclosedPorch 3SsnPorch ScreenPorch
PoolArea PoolQC Fence MiscFeature MiscVal MoSold
YrSold SaleType SaleCondition SalePrice

EDA (Exploratory Data Analysis)

We found that there were huge amounts of missing values in the columns (PoolQC, MiscFearture, Alley, Fence, MasVnrType, FireplaceQu, LotFrontage).

Features with Missing Values:

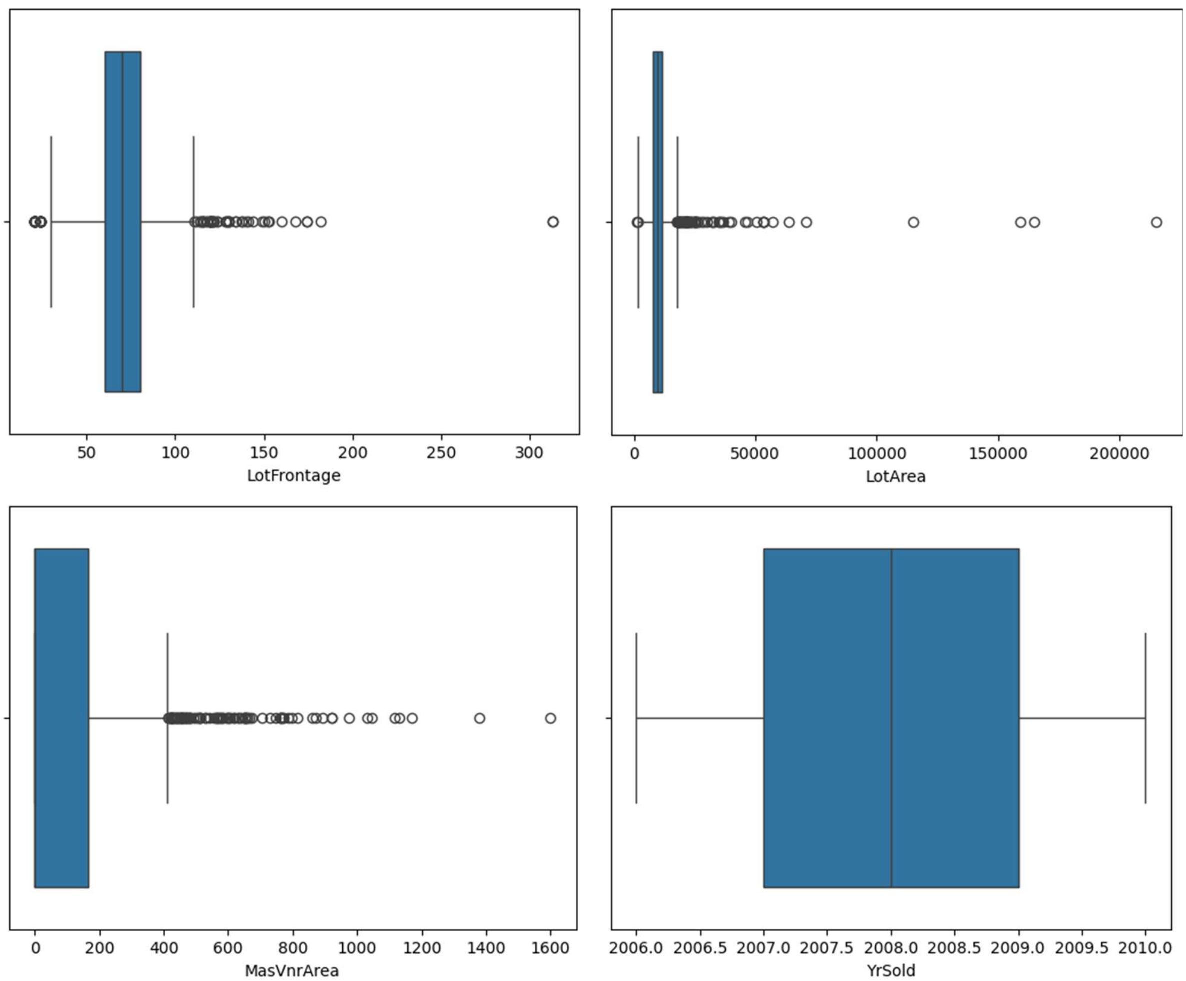
	Missing Count	Missing Percent
PoolQC	1453	99.520548
MiscFeature	1406	96.301370
Alley	1369	93.767123
Fence	1179	80.753425
MasVnrType	872	59.726027
FireplaceQu	690	47.260274
LotFrontage	259	17.739726
GarageType	81	5.547945
GarageYrBlt	81	5.547945
GarageFinish	81	5.547945
GarageQual	81	5.547945
GarageCond	81	5.547945
BsmtFinType2	38	2.602740
BsmtExposure	38	2.602740
BsmtFinType1	37	2.534247
BsmtCond	37	2.534247
BsmtQual	37	2.534247
MasVnrArea	8	0.547945
Electrical	1	0.068493

Figure 1: Missing Values

Data Processing

STEPS TAKEN

- We addressed some of the missing values by dropping the columns with the most missing values.
- We checked outliers by exploring graphs that visualize these outliers.
- We addressed outliers by using whisker function.



Example Plots/Graphs

Feature Engineering

STEPS TAKEN

- We created new features by combining features together to help simplify the data to make it less complex for the models.
- We also followed this step by dropping the columns used to create new features.
- Highlighted Categorial Features for model use.

ENCODING

- We Used One-Hot Encoding and Label Encoding

```
categorical_features = df.select_dtypes(include=['object', 'category']).columns

ordinal_features = []
one_hot_features = []

for feature in categorical_features:
    if df[feature].nunique() > 8:
        one_hot_features.append(feature)
    else:
        ordinal_features.append(feature)

if 'SalePrice' in df.columns:
    ordinal_mappings = {}

    for feature in ordinal_features:
        labels_ordered = df.groupby([feature])['SalePrice'].mean().sort_values().index
        ordinal_mappings[feature] = {k: i for i, k in enumerate(labels_ordered, 0)}
        df[feature] = df[feature].map(ordinal_mappings[feature])

else:
    for feature in ordinal_features:
        df[feature] = df[feature].map(ordinal_mappings.get(feature, {})).fillna(0)


df = pd.get_dummies(df, columns=one_hot_features, drop_first=True)
```

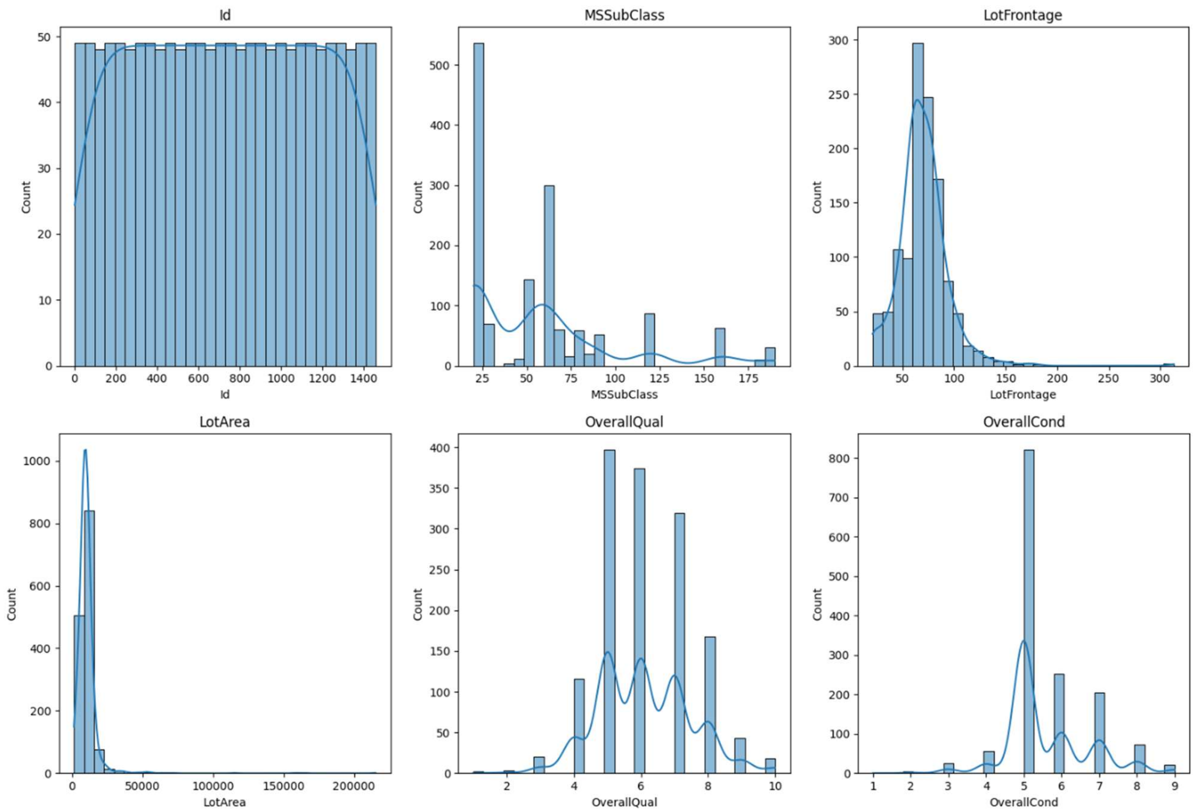
We applied One-Hot Encoding on high cardinality features and label encoding low cardinality features.

Feature Scaling

STEPS TAKEN

- Used Minmax Scaler.
- Test Split is 20%.

Regarding the use of Minmax Scaler, we explored the histograms of features, here are some examples of the graphs:




```python
for col in num_features:
    print(f"{col}: Skewness = {skew(df[col]):.2f}, Kurtosis = {kurtosis(df[col]):.2f}")
```

```python
for col in num_features:
    stat, p = shapiro(df[col].dropna())
    print(f"{col}: p-value = {p:.4f}")
```

We concluded from the graphs and the output of these 2 code snippets that the data doesn’t follow normal distribution, this forces us to not use standard scaler.

Models

LINEAR REGRESSION

```
reg_linear = LinearRegression()
reg_linear.fit(X_train_scaled, y_train)

train_score = reg_linear.score(X_train_scaled, y_train)
test_score = reg_linear.score(X_test_scaled, y_test)

print(f"Linear Train Score: {train_score:.4f}")
print(f"Linear Test Score: {test_score:.4f}")
```

```
Linear Train Score: 0.9211
Linear Test Score: 0.9176
```

Figure 2: Linear Regression

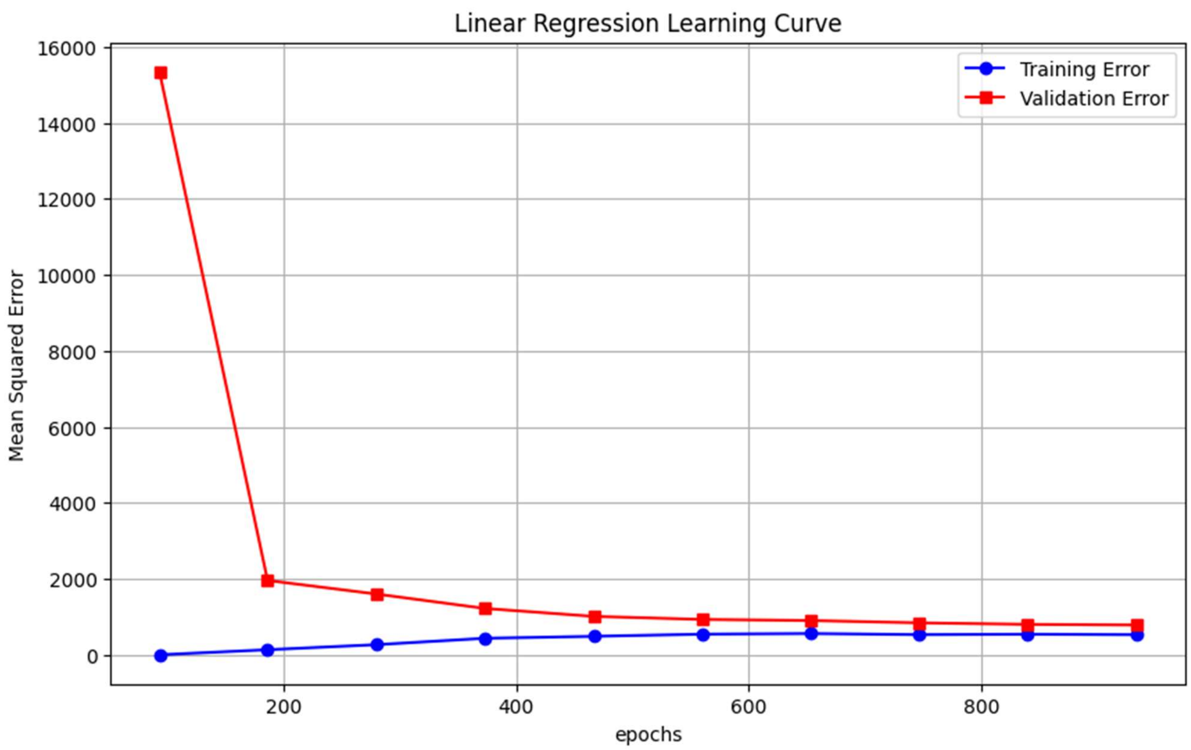


Figure 3: Linear Regression Learning Curve

RIDGE

```
param_grid = {"alpha": [0.01, 0.1, 1, 10, 100]}
ridge_grid = GridSearchCV(Ridge(fit_intercept=False), param_grid, cv=5, scoring="r2")
ridge_grid.fit(X_train_scaled, y_train)

best_ridge = ridge_grid.best_estimator_
print(f"Best Ridge Alpha: {ridge_grid.best_params_['alpha']}")
print(f"Best Ridge Train Score: {best_ridge.score(X_train_scaled, y_train):.4f}")
print(f"Best Ridge Test Score: {best_ridge.score(X_test_scaled, y_test):.4f}")
```

```
Best Ridge Alpha: 1
Best Ridge Train Score: 0.9200
Best Ridge Test Score: 0.9215
```

Figure 4: Ridge

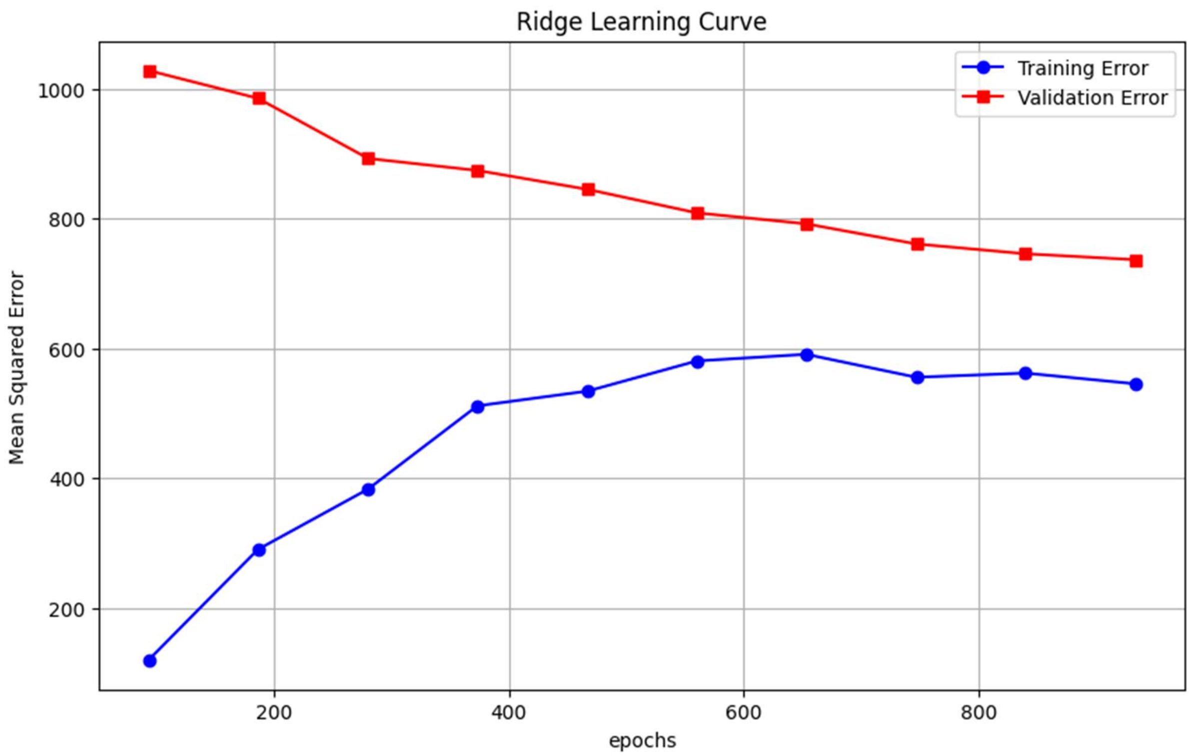


Figure 5: Ridge Learning Curve

LASSO

```
param_grid = {"alpha": [0.001, 0.01, 0.1, 1, 10]}
lasso_grid = GridSearchCV(Lasso(fit_intercept=False), param_grid, cv=5, scoring="r2")
lasso_grid.fit(X_train_scaled, y_train)

best_lasso = lasso_grid.best_estimator_
print(f"Best Lasso Alpha: {lasso_grid.best_params_['alpha']}")
print(f"Best Lasso Train Score: {best_lasso.score(X_train_scaled, y_train):.4f}")
print(f"Best Lasso Test Score: {best_lasso.score(X_test_scaled, y_test):.4f}")

Best Lasso Alpha: 0.1
Best Lasso Train Score: 0.9153
Best Lasso Test Score: 0.9248
```

Figure 6: Lasso

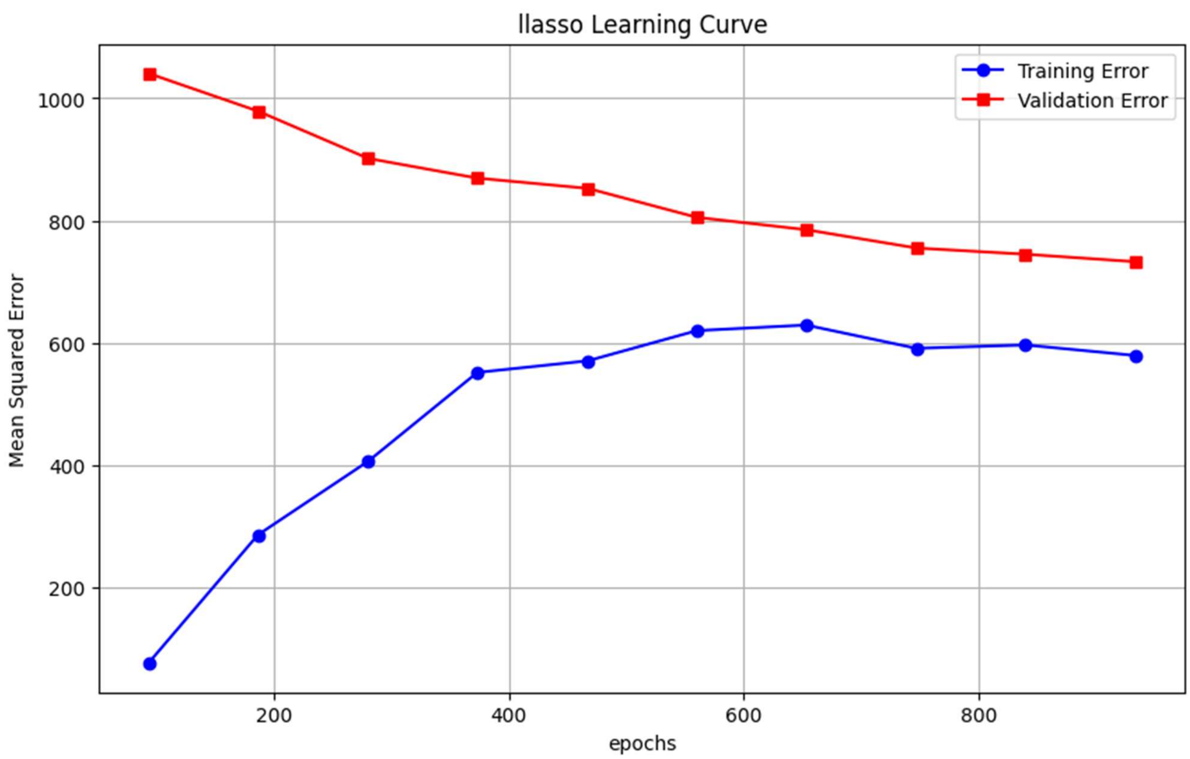


Figure 7: Lasso Learning Curve

Conclusion

Following the steps in the task description, we were able to conclude that the model with the highest accuracy with very little difference between the three, was linear regression, followed by Lasso and Ridge.