House Prices Regression Task - Beginner's Approach

Table of contents

Importing Libraries 1
Data Loading and Exploratory Data Analysis 2
Data Preprocessing 8
Model Training and Evaluation 9
Feature Importance (Random Forest) 11
Predictions on Test Data 12

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder

# Set random seed for reproducibility
np.random.seed(42)
```

This code imports necessary libraries. pandas and numpy are for data manipulation, matplotlib and seaborn for visualization, and scikit-learn modules for machine learning tasks.

Data Loading and Exploratory Data Analysis

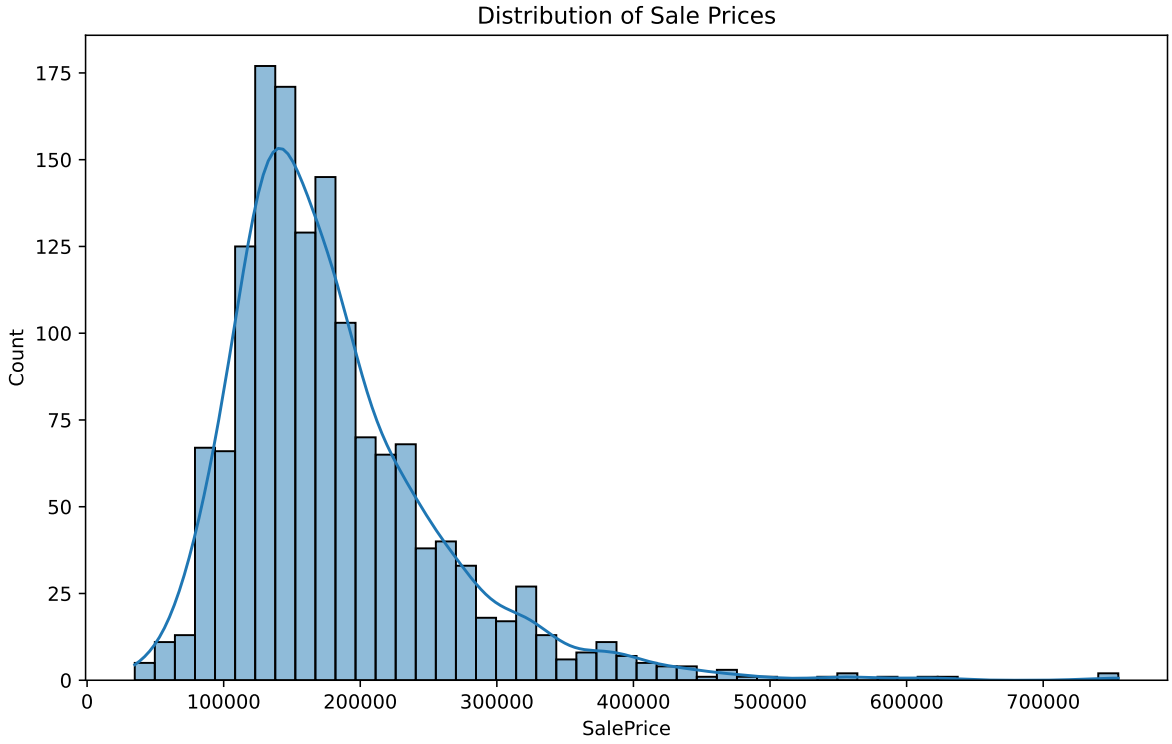
```
# Load the data
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")

# Display basic information about the dataset
print(train.info())

# Display summary statistics of numerical columns
print(train.describe())

# Plot distribution of target variable (SalePrice)
plt.figure(figsize=(10, 6))
sns.histplot(train["SalePrice"], kde=True)
plt.title("Distribution of Sale Prices")
plt.show()

# Scatter plot of GrLivArea vs SalePrice
plt.figure(figsize=(10, 6))
plt.scatter(train['GrLivArea'], train['SalePrice'], alpha=0.5)
plt.title('GrLivArea vs SalePrice')
plt.xlabel('GrLivArea (Above ground living area)')
plt.ylabel('SalePrice')
plt.show()

# Box plot of SalePrice by OverallQual
plt.figure(figsize=(12, 6))
sns.boxplot(x='OverallQual', y='SalePrice', data=train)
plt.title('SalePrice by Overall Quality')
plt.show()

# Correlation heatmap of numerical features
numeric_features = train.select_dtypes(include=[np.number])
plt.figure(figsize=(12, 10))
sns.heatmap(numeric_features.corr(), cmap='coolwarm', annot=False)
plt.title("Correlation Heatmap of Numerical Features")
plt.show()
```

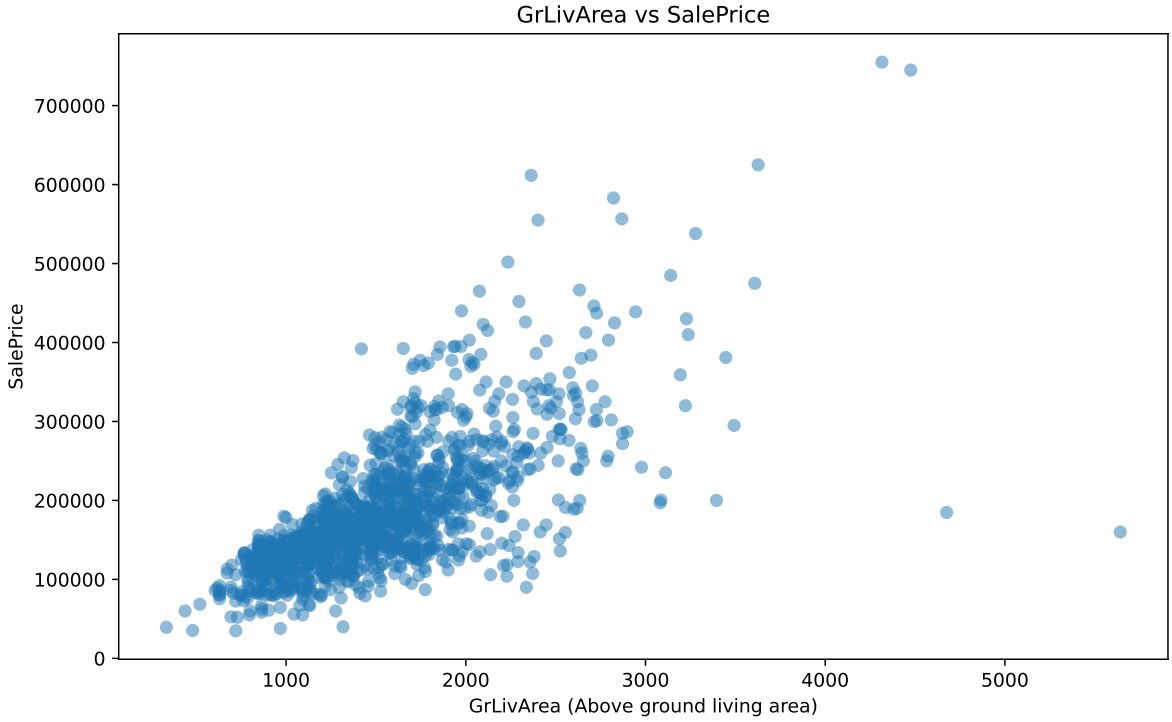
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
```

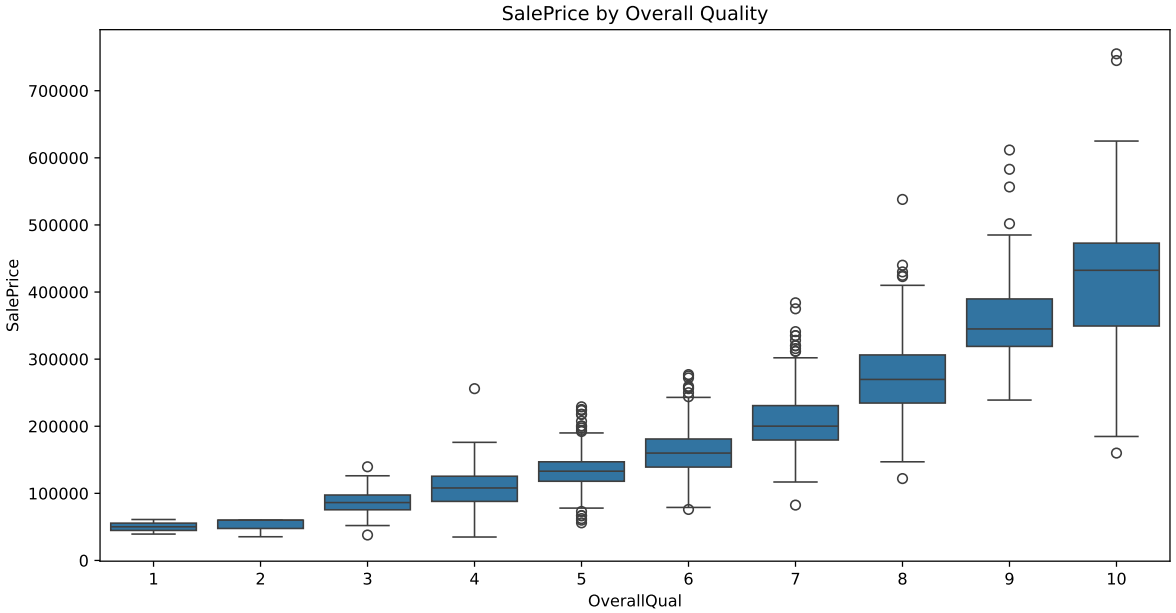
```
Data columns (total 81 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Id              1460 non-null   int64
 1   MSSubClass      1460 non-null   int64
 2   MSZoning        1460 non-null   object
 3   LotFrontage     1201 non-null   float64
 4   LotArea         1460 non-null   int64
 5   Street          1460 non-null   object
 6   Alley           91 non-null     object
 7   LotShape        1460 non-null   object
 8   LandContour     1460 non-null   object
 9   Utilities       1460 non-null   object
 10  LotConfig       1460 non-null   object
 11  LandSlope       1460 non-null   object
 12  Neighborhood    1460 non-null   object
 13  Condition1      1460 non-null   object
 14  Condition2      1460 non-null   object
 15  BldgType        1460 non-null   object
 16  HouseStyle      1460 non-null   object
 17  OverallQual     1460 non-null   int64
 18  OverallCond     1460 non-null   int64
 19  YearBuilt       1460 non-null   int64
 20  YearRemodAdd    1460 non-null   int64
 21  RoofStyle       1460 non-null   object
 22  RoofMatl        1460 non-null   object
 23  Exterior1st     1460 non-null   object
 24  Exterior2nd     1460 non-null   object
 25  MasVnrType      588 non-null    object
 26  MasVnrArea      1452 non-null   float64
 27  ExterQual       1460 non-null   object
 28  ExterCond       1460 non-null   object
 29  Foundation      1460 non-null   object
 30  BsmtQual        1423 non-null   object
 31  BsmtCond        1423 non-null   object
 32  BsmtExposure    1422 non-null   object
 33  BsmtFinType1    1423 non-null   object
 34  BsmtFinSF1      1460 non-null   int64
 35  BsmtFinType2    1422 non-null   object
 36  BsmtFinSF2      1460 non-null   int64
 37  BsmtUnfSF       1460 non-null   int64
 38  TotalBsmtSF     1460 non-null   int64
 39  Heating         1460 non-null   object
```

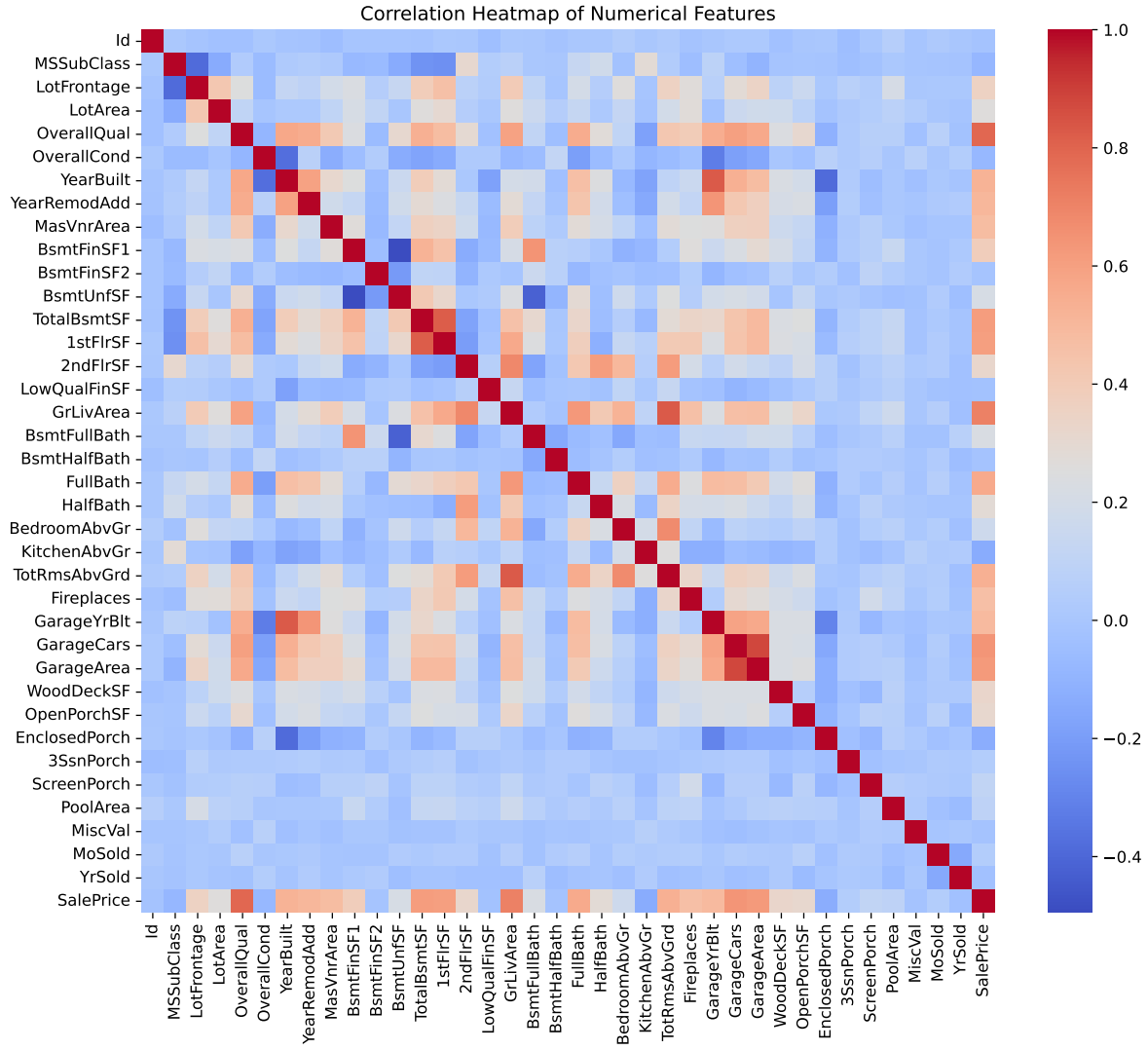
```
 40  HeatingQC      1460 non-null   object
 41  CentralAir     1460 non-null   object
 42  Electrical     1459 non-null   object
 43  1stFlrSF       1460 non-null   int64
 44  2ndFlrSF       1460 non-null   int64
 45  LowQualFinSF   1460 non-null   int64
 46  GrLivArea      1460 non-null   int64
 47  BsmtFullBath   1460 non-null   int64
 48  BsmtHalfBath   1460 non-null   int64
 49  FullBath       1460 non-null   int64
 50  HalfBath       1460 non-null   int64
 51  BedroomAbvGr   1460 non-null   int64
 52  KitchenAbvGr   1460 non-null   int64
 53  KitchenQual    1460 non-null   object
 54  TotRmsAbvGrd   1460 non-null   int64
 55  Functional     1460 non-null   object
 56  Fireplaces     1460 non-null   int64
 57  FireplaceQu    770 non-null    object
 58  GarageType     1379 non-null   object
 59  GarageYrBlt    1379 non-null   float64
 60  GarageFinish   1379 non-null   object
 61  GarageCars     1460 non-null   int64
 62  GarageArea     1460 non-null   int64
 63  GarageQual     1379 non-null   object
 64  GarageCond     1379 non-null   object
 65  PavedDrive     1460 non-null   object
 66  WoodDeckSF     1460 non-null   int64
 67  OpenPorchSF    1460 non-null   int64
 68  EnclosedPorch  1460 non-null   int64
 69  3SsnPorch      1460 non-null   int64
 70  ScreenPorch    1460 non-null   int64
 71  PoolArea       1460 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          281 non-null    object
 74  MiscFeature    54 non-null     object
 75  MiscVal        1460 non-null   int64
 76  MoSold         1460 non-null   int64
 77  YrSold         1460 non-null   int64
 78  SaleType       1460 non-null   object
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

```
None
                Id   MSSubClass  LotFrontage        LotArea  OverallQual  \
count  1460.000000  1460.000000  1201.000000    1460.000000  1460.000000
mean    730.500000    56.897260    70.049958   10516.828082     6.099315
std     421.610009    42.300571    24.284752    9981.264932     1.382997
min       1.000000    20.000000    21.000000    1300.000000     1.000000
25%     365.750000    20.000000    59.000000    7553.500000     5.000000
50%     730.500000    50.000000    69.000000    9478.500000     6.000000
75%    1095.250000    70.000000    80.000000   11601.500000     7.000000
max    1460.000000   190.000000   313.000000  215245.000000    10.000000

       OverallCond    YearBuilt  YearRemodAdd   MasVnrArea   BsmtFinSF1  ...  \
count  1460.000000  1460.000000   1460.000000  1452.000000  1460.000000  ...
mean      5.575342  1971.267808   1984.865753   103.685262   443.639726  ...
std       1.112799    30.202904     20.645407   181.066207   456.098091  ...
min       1.000000  1872.000000   1950.000000     0.000000     0.000000  ...
25%       5.000000  1954.000000   1967.000000     0.000000     0.000000  ...
50%       5.000000  1973.000000   1994.000000     0.000000   383.500000  ...
75%       6.000000  2000.000000   2004.000000   166.000000   712.250000  ...
max       9.000000  2010.000000   2010.000000  1600.000000  5644.000000  ...

        WoodDeckSF  OpenPorchSF  EnclosedPorch    3SsnPorch  ScreenPorch  \
count  1460.000000  1460.000000    1460.000000  1460.000000  1460.000000
mean     94.244521    46.660274      21.954110     3.409589    15.060959
std     125.338794    66.256028      61.119149    29.317331    55.757415
min       0.000000     0.000000       0.000000     0.000000     0.000000
25%       0.000000     0.000000       0.000000     0.000000     0.000000
50%       0.000000    25.000000       0.000000     0.000000     0.000000
75%     168.000000    68.000000       0.000000     0.000000     0.000000
max     857.000000   547.000000     552.000000   508.000000   480.000000

          PoolArea       MiscVal       MoSold       YrSold      SalePrice
count  1460.000000   1460.000000  1460.000000  1460.000000    1460.000000
mean      2.758904     43.489041     6.321918  2007.815753  180921.195890
std      40.177307    496.123024     2.703626     1.328095   79442.502883
min       0.000000      0.000000     1.000000  2006.000000   34900.000000
25%       0.000000      0.000000     5.000000  2007.000000  129975.000000
50%       0.000000      0.000000     6.000000  2008.000000  163000.000000
75%       0.000000      0.000000     8.000000  2009.000000  214000.000000
max     738.000000  15500.000000    12.000000  2010.000000  755000.000000

[8 rows x 38 columns]
```


Distribution of Sale Prices
Count
SalePrice
0
25
50
75
100
125
150
175
0
100000
200000
300000
400000
500000
600000
700000


GrLivArea vs SalePrice
SalePrice
GrLivArea (Above ground living area)
0
100000
200000
300000
400000
500000
600000
700000
1000
2000
3000
4000
5000


SalePrice by Overall Quality
SalePrice
0
100000
200000
300000
400000
500000
600000
700000
1
2
3
4
5
6
7
8
9
10
OverallQual



This section loads the data and performs basic exploratory data analysis. It includes visualizations of the target variable distribution, relationship between key features, and correlation between numerical features.

Data Preprocessing

```
def preprocess_data(df):
    # Handle missing values
    for col in df.columns:
        if df[col].dtype != "object":
```

```
            df[col] = df[col].fillna(df[col].median())
        else:
            df[col] = df[col].fillna(df[col].mode()[0])

    # Encode categorical variables
    le = LabelEncoder()
    for col in df.select_dtypes(include=["object"]).columns:
        df[col] = le.fit_transform(df[col].astype(str))

    return df

# Preprocess train and test data
X = preprocess_data(train.drop("SalePrice", axis=1))
y = train["SalePrice"]
test_processed = preprocess_data(test)

print("Processed data shape:", X.shape)
```

```
Processed data shape: (1460, 80)
```

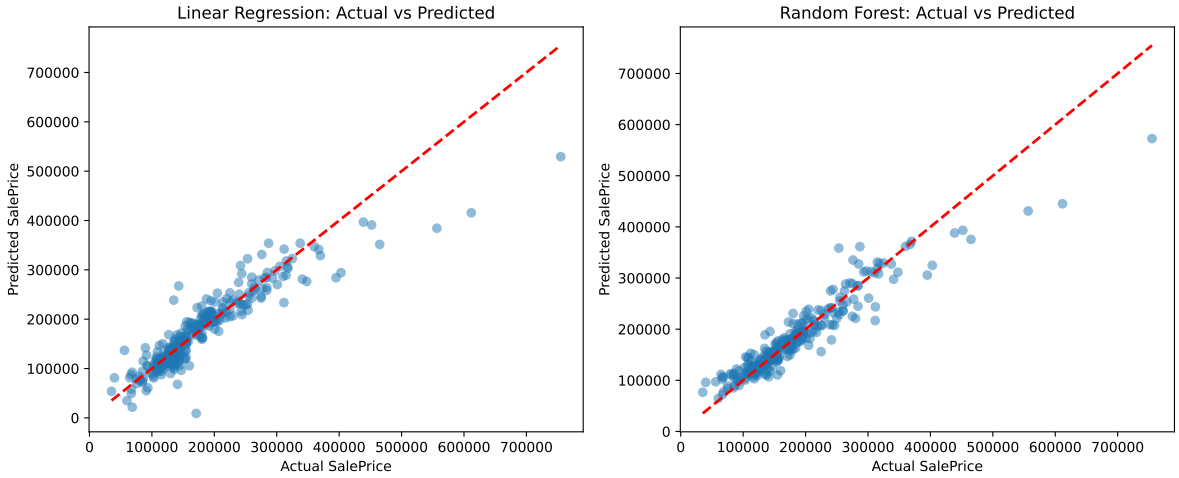
This code preprocesses the data by handling missing values and encoding categorical variables. It uses median imputation for numerical features and mode imputation for categorical features.

Model Training and Evaluation

```
# Split the data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# Train and evaluate Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_train_pred = lr_model.predict(X_train)
lr_val_pred = lr_model.predict(X_val)

print("Linear Regression Results:")
print(f"Train R2 Score: {r2_score(y_train, lr_train_pred):.4f}")
print(f"Validation R2 Score: {r2_score(y_val, lr_val_pred):.4f}")
print(f"Validation RMSE: {np.sqrt(mean_squared_error(y_val, lr_val_pred)):.4f}")
```

```
# Train and evaluate Random Forest
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_train_pred = rf_model.predict(X_train)
rf_val_pred = rf_model.predict(X_val)

print("\nRandom Forest Results:")
print(f"Train R2 Score: {r2_score(y_train, rf_train_pred):.4f}")
print(f"Validation R2 Score: {r2_score(y_val, rf_val_pred):.4f}")
print(f"Validation RMSE: {np.sqrt(mean_squared_error(y_val, rf_val_pred)):.4f}")

# Plot actual vs predicted values for both models
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(y_val, lr_val_pred, alpha=0.5)
plt.plot([y_val.min(), y_val.max()], [y_val.min(), y_val.max()], 'r--', lw=2)
plt.xlabel("Actual SalePrice")
plt.ylabel("Predicted SalePrice")
plt.title("Linear Regression: Actual vs Predicted")

plt.subplot(1, 2, 2)
plt.scatter(y_val, rf_val_pred, alpha=0.5)
plt.plot([y_val.min(), y_val.max()], [y_val.min(), y_val.max()], 'r--', lw=2)
plt.xlabel("Actual SalePrice")
plt.ylabel("Predicted SalePrice")
plt.title("Random Forest: Actual vs Predicted")

plt.tight_layout()
plt.show()
```

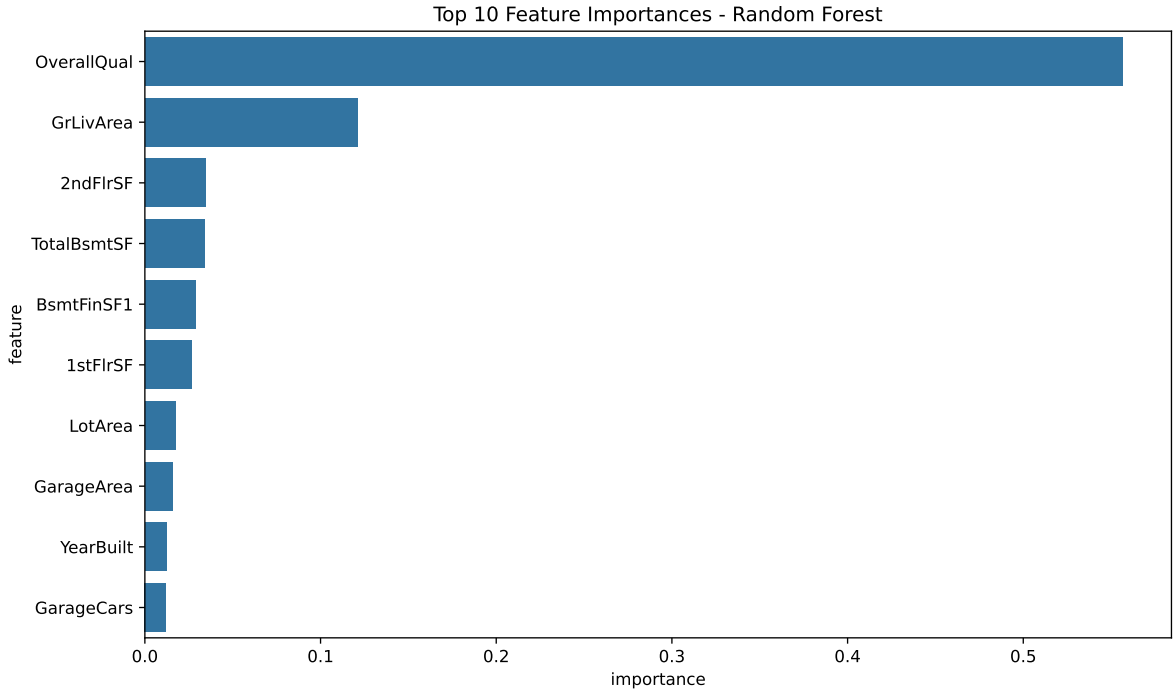
```
Linear Regression Results:
Train R2 Score: 0.8578
Validation R2 Score: 0.8371
Validation RMSE: 35344.2071

Random Forest Results:
Train R2 Score: 0.9793
Validation R2 Score: 0.8903
Validation RMSE: 29001.5337
```



This section trains and evaluates two models: Linear Regression and Random Forest. It calculates R2 score and RMSE for both models and visualizes their performance using actual vs predicted plots.

Feature Importance (Random Forest)

```
# Get feature importances from Random Forest
importances = rf_model.feature_importances_
feature_imp = pd.DataFrame({'feature': X.columns, 'importance': importances})
feature_imp = feature_imp.sort_values('importance', ascending=False).head(10)

# Plot feature importances
plt.figure(figsize=(10, 6))
sns.barplot(x='importance', y='feature', data=feature_imp)
plt.title('Top 10 Feature Importances - Random Forest')
plt.tight_layout()
plt.show()
```



This code extracts and visualizes the top 10 most important features according to the Random Forest model.

Predictions on Test Data

```
# Make predictions on the test data using Random Forest
test_predictions = rf_model.predict(test_processed)

# Create submission file
submission = pd.DataFrame({'Id': test['Id'], 'SalePrice': test_predictions})
submission.to_csv('submission.csv', index=False)
print("Submission file created.")
```

```
Submission file created.
```

This final section uses the Random Forest model to make predictions on the test data and creates a submission file.