# House Price Prediction Project Report

# House Price Prediction Project Report

## Introduction

This project aimed to predict house prices based on a dataset with 79 features describing various attributes of residential homes in Ames, Iowa. The goal was to build a model to predict the sale price of each home using advanced regression techniques.

## Objectives

1. Analyze the dataset to understand the key factors influencing house prices.
2. Handle missing data and perform feature engineering.
3. Apply machine learning models, including Linear Regression and Random Forest, to predict house prices.
4. Optimize model performance through hyperparameter tuning.

## Steps Followed

### 1. Data Loading and Exploration

We began by loading the training and test datasets using pandas. The dataset contained 81 columns in the training set, with `SalePrice` as the target variable. We inspected the data to identify missing values and understand the overall structure.

```python
import pandas as pd
train_data = pd.read_csv('train.csv')
```

test_data = pd.read_csv('test.csv')

print(train_data.info())

### 2. Handling Missing Data

Several columns had missing data. We handled missing values using strategies such as:

- Filling numerical columns with the median or zero.

- Filling categorical columns with the most frequent value or "None".

- Dropping columns with excessive missing data.

```python
train_data['LotFrontage'].fillna(train_data['LotFrontage'].median(), inplace=True)

train_data['Alley'].fillna("None", inplace=True)

train_data.drop(['PoolQC', 'MiscFeature'], axis=1, inplace=True)
```

### 3. Exploratory Data Analysis (EDA)

We analyzed the distribution of `SalePrice`, which was right-skewed. A log transformation was applied to normalize it. We also examined the correlations between `SalePrice` and key features.

```python
import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

train_data['SalePrice'] = np.log(train_data['SalePrice'])
```

sns.histplot(train_data['SalePrice'], kde=True)

plt.show()

~~~

4. Feature Selection

We selected key features based on their correlation with `SalePrice`. The top features included `OverallQual`, `GrLivArea`, `GarageCars`, and others.

5. Model Building and Evaluation

We built two models: Linear Regression and Random Forest. The performance was evaluated using Root Mean Squared Error (RMSE) on both the log-transformed and original scale.

Linear Regression

```python
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error

X = train_data[['OverallQual', 'GrLivArea', 'GarageCars', 'GarageArea', 'TotalBsmtSF', '1stFlrSF', 'FullBath', 'TotRmsAbvGrd', 'YearBuilt']]

y = train_data['SalePrice']

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()

model.fit(X_train, y_train)
~~~

y_pred = model.predict(X_val)
rmse = np.sqrt(mean_squared_error(y_val, y_pred))
print(f'Linear Regression RMSE: {rmse}')

#### Random Forest

```python
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(n_estimators=200, max_depth=None, min_samples_split=10, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_val)

rmse_rf = np.sqrt(mean_squared_error(y_val, y_pred_rf))
print(f'Random Forest RMSE: {rmse_rf}')
```

### 6. Hyperparameter Tuning

We used GridSearchCV to tune the Random Forest model, achieving the best parameters and improved performance.

```python
from sklearn.model_selection import GridSearchCV

param_grid = {
```

    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10]
}

rf_grid_search = GridSearchCV(RandomForestRegressor(random_state=42), param_grid, cv=3, n_jobs=-1)
rf_grid_search.fit(X_train, y_train)
best_rf_model = rf_grid_search.best_estimator_

### 7. Predictions and Submission

Finally, we used the tuned Random Forest model to predict `SalePrice` for the test data and created a submission file for Kaggle.

```python
test_predictions = best_rf_model.predict(test_data[selected_features])
submission = pd.DataFrame({'Id': test_data['Id'], 'SalePrice': np.exp(test_predictions)})
submission.to_csv('submission.csv', index=False)
```

## Conclusion

This project demonstrated how to build and optimize machine learning models for predicting house prices using Linear Regression and Random Forest. Through data preprocessing, feature selection, and hyperparameter tuning, we improved the models performance and prepared the predictions for Kaggle submission.