HOUSE PRICE PREDICTION

The objective of this project is to develop a robust predictive model that accurately estimates the sale prices of residential properties based on various descriptive features. The dataset comprises numerous attributes describing different aspects of residential properties. The aim is to create a regression model capable of predicting the sale prices of houses given their characteristics.

DATASET:

The dataset provided contains comprehensive information about residential properties. It's specifically tailored for predicting house sale prices based on numerous descriptive features associated with these properties. Here's a detailed description of the dataset columns:

- **SalePrice** - the property's sale price in dollars. This is the target variable that you're trying to predict.
- **MSSubClass**: The building class
- **MSZoning**: The general zoning classification
- **LotFrontage**: Linear feet of street connected to property
- **LotArea**: Lot size in square feet
- **Street**: Type of road access
- **Alley**: Type of alley access
- **LotShape**: General shape of property
- **LandContour**: Flatness of the property
- **Utilities**: Type of utilities available
- **LotConfig**: Lot configuration
- **LandSlope**: Slope of property
- **Neighborhood**: Physical locations within Ames city limits
- **Condition1**: Proximity to main road or railroad
- **Condition2**: Proximity to main road or railroad (if a second is present)
- **BldgType**: Type of dwelling
- **HouseStyle**: Style of dwelling
- **OverallQual**: Overall material and finish quality
- **OverallCond**: Overall condition rating
- **YearBuilt**: Original construction date
- **YearRemodAdd**: Remodel date
- **RoofStyle**: Type of roof
- **RoofMatl**: Roof material
- **Exterior1st**: Exterior covering on house
- **Exterior2nd**: Exterior covering on house (if more than one material)
- **MasVnrType**: Masonry veneer type
- **MasVnrArea**: Masonry veneer area in square feet
- **ExterQual**: Exterior material quality
- **ExterCond**: Present condition of the material on the exterior
- **Foundation**: Type of foundation
- **BsmtQual**: Height of the basement
- **BsmtCond**: General condition of the basement

- **BsmtExposure**: Walkout or garden level basement walls
- **BsmtFinType1**: Quality of basement finished area
- **BsmtFinSF1**: Type 1 finished square feet
- **BsmtFinType2**: Quality of second finished area (if present)
- **BsmtFinSF2**: Type 2 finished square feet
- **BsmtUnfSF**: Unfinished square feet of basement area
- **TotalBsmtSF**: Total square feet of basement area
- **Heating**: Type of heating
- **HeatingQC**: Heating quality and condition
- **CentralAir**: Central air conditioning
- **Electrical**: Electrical system
- **1stFlrSF**: First Floor square feet
- **2ndFlrSF**: Second floor square feet
- **LowQualFinSF**: Low quality finished square feet (all floors)
- **GrLivArea**: Above grade (ground) living area square feet
- **BsmtFullBath**: Basement full bathrooms
- **BsmtHalfBath**: Basement half bathrooms
- **FullBath**: Full bathrooms above grade
- **HalfBath**: Half baths above grade
- **Bedroom**: Number of bedrooms above basement level
- **Kitchen**: Number of kitchens
- **KitchenQual**: Kitchen quality
- **TotRmsAbvGrd**: Total rooms above grade (does not include bathrooms)
- **Functional**: Home functionality rating
- **Fireplaces**: Number of fireplaces
- **FireplaceQu**: Fireplace quality
- **GarageType**: Garage location
- **GarageYrBlt**: Year garage was built
- **GarageFinish**: Interior finish of the garage
- **GarageCars**: Size of garage in car capacity
- **GarageArea**: Size of garage in square feet
- **GarageQual**: Garage quality
- **GarageCond**: Garage condition
- **PavedDrive**: Paved driveway
- **WoodDeckSF**: Wood deck area in square feet
- **OpenPorchSF**: Open porch area in square feet
- **EnclosedPorch**: Enclosed porch area in square feet
- **3SsnPorch**: Three season porch area in square feet
- **ScreenPorch**: Screen porch area in square feet
- **PoolArea**: Pool area in square feet
- **PoolQC**: Pool quality
- **Fence**: Fence quality
- **MiscFeature**: Miscellaneous feature not covered in other categories
- **MiscVal**: $Value of miscellaneous feature
- **MoSold**: Month Sold
- **YrSold**: Year Sold
- **SaleType**: Type of sale
- **SaleCondition**: Condition of sale

STEPS TAKEN:

1. Importing Libraries
2. Data Loading and Initial Examination
3. Data Cleaning
4. Exploratory Data Analysis(EDA)
5. Data Preprocessing
6. Train/Test Split
7. Model Building and Evaluation
8. Hyperparameter Tuning
9. Final Model Assessment

DATA LOADING AND INITIAL EXAMINATION:

The dataset is loaded using Pandas' **read_csv** function. **head(), shape**, **info()**, and **describe()** functions are used to get an initial glimpse of the data's structure, the number of rows and columns, data types, and basic statistics like mean, standard deviation, etc.

DATA CLEANING:

1. **Checking for missing values:**
   Missing values are identified using **data.isnull().sum()** which counts the number of missing values in each column.
   The percentage of missing values per column is calculated to determine columns with substantial missing data.
   Columns with missing values exceeding a certain threshold (80% in this case) are dropped using **data = data.drop((missing[missing['Total'] > 81]).index, axis=1)**.
   For columns with fewer missing values, strategies such as mode imputation for categorical columns and sampling imputation for numerical columns (**fillna() method**) are employed to replace missing values.
2. **Removing columns with weak correlation:**
   Correlation coefficients between features and the target variable (SalePrice) are computed using
   corr_matrix['SalePrice'].apply(abs).sort_values(ascending=False).
   Columns with weak correlation to the target ('SalePrice') are dropped using **data = data.drop(correlations.iloc[21:, 0].values, axis=1)**. Here, the weakest correlated features are removed.
3. **Outliers Handling:**
   Outliers are identified using the IQR method within a loop that iterates through numerical columns. It uses **sns.boxplot** to visualize the distribution of each numerical feature and detect outliers beyond the IQR boundaries.
   For each numerical column, the IQR (Interquartile Range) is calculated to determine the boundaries of potential outliers: **l_bound = Q1 - (1.5 * IQR)** and **u_bound = Q3 + (1.5 * IQR).**

Rows containing values outside these boundaries are filtered and kept in the dataset: **data = data[(data[col] >= l_bound) & (data[col] <= u_bound)].**

EXPLORATORY DATA ANALYSIS(EDA):

- Boxplots are used to identify outliers.
- Correlation matrix and heatmap are used to understand feature relationships
- Other visualizations such as Histogram of SalePrice, Countplot of OverallQual, Scatterplot of GrLivArea vs. SalePrice, Barplot of Neighborhood vs. SalePrice, Barplot of BldgType vs. SalePrice, and Countplot of YearBuilt are also plotted to better understand the dataset.

DATA PREPROCESSING:

Encoding Categorical Variables:

Label Encoding: Categorical columns were transformed into numerical format using **LabelEncoder()** from scikit-learn. Each unique category within a column was assigned a numerical label.

Scaling Numerical Data: Numerical columns were scaled using **StandardScaler()** from scikit-learn to bring them to a standard scale (mean = 0, standard deviation = 1). This ensures that all features contribute equally to the model training process, preventing features with larger scales from dominating the learning process.

TRAIN/TEST SPLIT:

The dataset is split into training and testing sets using **train_test_split** from scikit-learn.

MODEL BUILDING AND EVALUATION:

Three regression models are used:

- **Linear Regression**: Built using **LinearRegression()** from scikit-learn.
- **Random Forest Regressor**: Constructed using **RandomForestRegressor()** from scikit-learn.
- **XGBoost Regressor**: Constructed using **XGBRegressor()** from the XGBoost library.

For each model, the following evaluation metrics are computed on the test set:

Mean Squared Error (MSE): Measures the average squared difference between the predicted values and the actual values.

Root Mean Squared Error (RMSE): Represents the square root of the MSE, giving an interpretable metric in the same unit as the target variable.

Mean Absolute Error (MAE): Measures the average absolute differences between the predicted and actual values.

R-squared (R2): Indicates the proportion of variance in the target variable explained by the model.

Interpretation of the results obtained on evaluating these models:

Random Forest Regressor and Linear Regression performed relatively similarly in terms of accuracy, with Random Forest slightly outperforming in some metrics.

XGBoost Regressor showed slightly higher errors compared to the other models but still performed reasonably well.

R-squared values for all models are around 0.87, indicating that approximately 87% of the variance in the target variable is explained by the models.

RMSE values are in the range of approximately 17,776 to 18,231, indicating the average prediction error of the models in dollars.

Based on these results, Random Forest Regressor is selected due to its slightly better performance for hyperparameter tuning.

HYPERPARAMETER TUNING:

RandomForestRegressor is chosen as the model to tune its hyperparameters.

A hyperparameter grid is created, specifying the range of values to explore for different hyperparameters:

```
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

GridSearchCV is employed, which performs an exhaustive search over the specified hyperparameter values.

FINAL MODEL ASSESMENT:

After performing the hyperparameter tuning for the RandomForestRegressor model:

The best parameters selected based on the search are:

```
Best Parameters: {'max_depth': 20, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 300}
```

The best score achieved after tuning (negative mean squared error) is:

```
Best Score: 352815443.261397
```

These results indicate that, among the specified hyperparameter combinations, the RandomForestRegressor model performs best when using the parameters mentioned above, resulting in the lowest mean squared error on the cross-validated data.

CONCLUSION:

In this house price prediction project, an extensive analysis of the dataset was conducted to create a robust predictive model. The dataset initially underwent thorough data cleaning, handling missing values, outliers, and correlation analysis, ensuring data quality for modeling. Exploratory Data Analysis (EDA) provided insights into feature distributions and relationships, guiding subsequent preprocessing steps such as label encoding and scaling.

Three models-Linear Regression, RandomForestRegressor, and XGBoost Regressor—were evaluated using key metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R-squared. The models exhibited promising performance, with RandomForestRegressor showcasing the most consistent and competitive results.

Hyperparameter tuning was applied to the RandomForestRegressor model using GridSearchCV, optimizing its parameters and further enhancing its predictive capability. The best-tuned model demonstrated notable improvements in performance metrics, displaying enhanced accuracy and generalization ability.