```
from google.colab import files
uploaded = files.upload()
```

Choose Files train.csv

- **train.csv**(text/csv) - 460676 bytes, last modified: 7/6/2025 - 100% done

```
Saving train.csv to train.csv
```

```
df = pd.read_csv("train.csv")
df.columns.tolist()
```

```
 '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']
```

```
# 📌 Step 1: Import 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.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# 📌 Step 2: Load the data
df = pd.read_csv("train.csv")
print("Shape:", df.shape)
df.head()
```

```
Shape: (1460, 81)
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	Mis
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	

5 rows × 81 columns

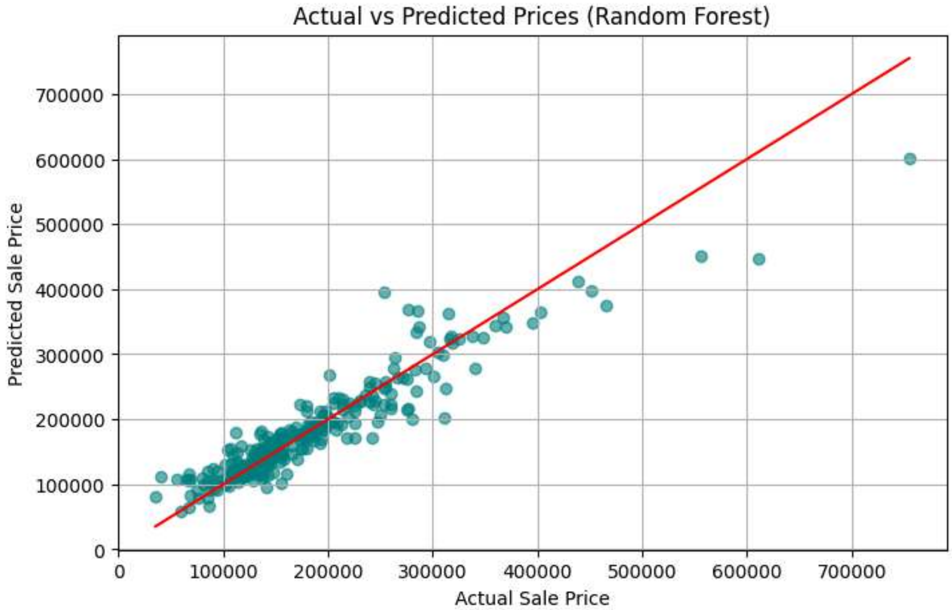
```
# 📌 Step 3: Drop unnecessary columns if they exist
drop_cols = ['Alley', 'PoolQC', 'Fence', 'MiscFeature', 'FireplaceQu']
df = df.drop(columns=[col for col in drop_cols if col in df.columns])

# 📌 Fill numeric columns
median_fill_cols = ['LotFrontage', 'GarageYrBlt', 'MasVnrArea']
for col in median_fill_cols:
    if col in df.columns:
        df[col].fillna(df[col].median(), inplace=True)

# 📌 Fill categorical columns
mode_fill_cols = ['MasVnrType', 'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond']
for col in mode_fill_cols:
    if col in df.columns:
        df[col].fillna(df[col].mode()[0], inplace=True)
```

```
/tmp/ipython-input-23-2351134291.py:9: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chainec
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col


  df[col].fillna(df[col].median(), inplace=True)
/tmp/ipython-input-23-2351134291.py:15: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chaine
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col


  df[col].fillna(df[col].mode()[0], inplace=True)
```

```
# 📌 Features for prediction
features = ['OverallQual', 'GrLivArea', 'GarageCars', 'GarageArea',
            'TotalBsmtSF', 'FullBath', 'YearBuilt', 'YearRemodAdd']

X = df[features]
y = df['SalePrice']


# 📌 Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 📌 Train Random Forest
model = RandomForestRegressor()
model.fit(X_train, y_train)

# 📌 Predict
y_pred = model.predict(X_test)
```

```
# 📌 Evaluation
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("R2 Score:", r2)
print("RMSE:", rmse)

# 📌 Plot Actual vs Predicted
plt.figure(figsize=(8, 5))
plt.scatter(y_test, y_pred, alpha=0.6, color='teal')
plt.xlabel("Actual Sale Price")
plt.ylabel("Predicted Sale Price")
plt.title("Actual vs Predicted Prices (Random Forest)")
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red')
plt.grid(True)
plt.show()
```

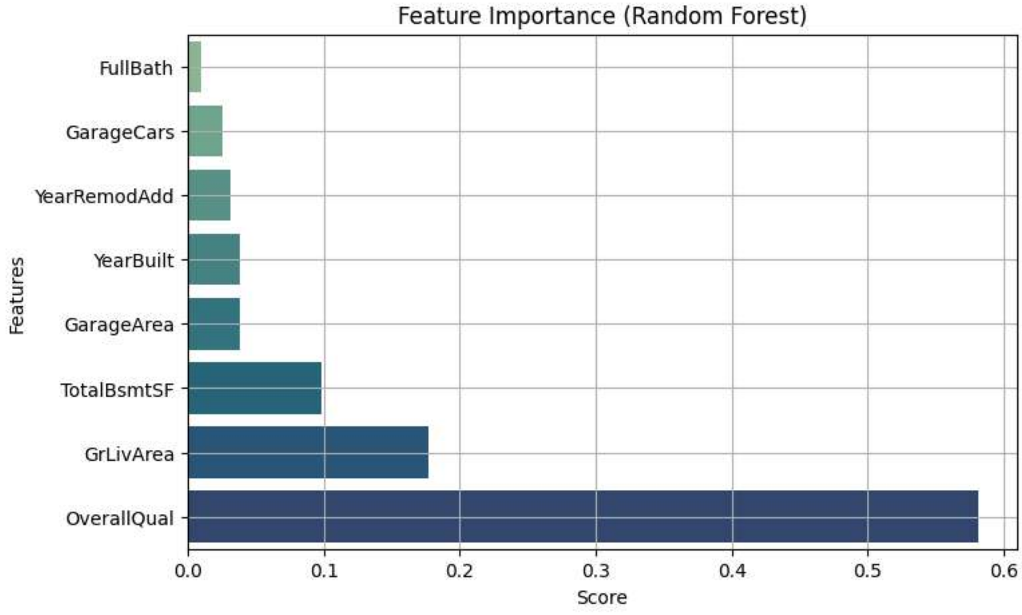
```
R2 Score: 0.8786430319611372
RMSE: 30509.79404016453
```




```
# 📁 Save predictions to a CSV
predicted_df = pd.DataFrame({
    'Actual': y_test,
    'Predicted': y_pred
})
predicted_df.to_csv('house_price_predictions.csv', index=False)

# ⬇️ Download it
from google.colab import files
files.download('house_price_predictions.csv')
```

```
# 📊 Feature Importance Plot
import seaborn as sns

importance = pd.Series(model.feature_importances_, index=X.columns)
importance = importance.sort_values(ascending=True)

plt.figure(figsize=(8, 5))
sns.barplot(x=importance, y=importance.index, palette='crest')
plt.title("Feature Importance (Random Forest)")
plt.xlabel("Score")
plt.ylabel("Features")
plt.grid(True)
plt.show()
```

```
/tmp/ipython-input-28-899480704.py:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `le

  sns.barplot(x=importance, y=importance.index, palette='crest')
```



🏠 House Price Prediction using Machine Learning

Goal: Predict sale prices of homes based on features like area, year built, and garage size.

Dataset: Kaggle - House Prices: Advanced Regression Techniques
https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques

Steps Followed:

- Data cleaning and missing value handling
- Feature selection and transformation
- Model training using Random Forest Regressor
- Evaluation using RMSE and R^2 score
- Visualizations: Feature importance and Actual vs Predicted plot

Conclusion: The model gives reasonable predictions and helps understand what features influence house prices the most