```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remou
```

```
import pandas as pd
```

```
df = pd.read_csv("/content/drive/MyDrive/House price prediction/house_price_pred_dummies(ready_to_split).csv")
df.head()
```

	BsmtFullBath	BsmtHalfBath	FullBath	HalfBath	BedroomAbvGr	KitchenAbvGr	tot
0	1.0	0.0	2.0	1.0	3.0	1.0	
1	0.0	1.0	2.0	0.0	3.0	1.0	
2	1.0	0.0	2.0	1.0	3.0	1.0	
3	1.0	0.0	1.0	0.0	3.0	1.0	
4	1.0	0.0	2.0	1.0	4.0	1.0	

5 rows × 165 columns

```
df.shape
```

```
(1102, 165)
```

```
## shuffle
from sklearn.utils import shuffle
```

```
clean = shuffle(df)
```

```
## reset indexes
clean.reset_index(inplace=True, drop=True)
```

```
from sklearn.model_selection import train_test_split
X = clean.drop('SalePrice', axis = 1)
y = clean['SalePrice']
```

```
y.head()
```

```
0    129000.0
1    227875.0
2    115000.0
3    155000.0
4    139950.0
Name: SalePrice, dtype: float64
```

```
x_train, x_test, y_train, y_test=train_test_split(X, y, test_size = 0.2)
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(x_train)

x_train= scaler.transform(x_train)
x_test= scaler.transform(x_test)
```

```
import numpy as np
```

We found that there're 4 models that contains the highest values

Model: KNN

- Train MAE: 17427.112145289444
- Test MAE: 21799.318552036202
- Train RMSE: 24068.9419386175
- Test RMSE: 31562.13157491167
- Train R2: 0.8095508064637456
- Test R2: 0.713117490113877

Model: DT

- Train MAE: 11.91827468785471
- Test MAE: 25576.945701357465
- Train RMSE: 192.4364364298879
- Test RMSE: 34853.38263120388
- Train R2: 0.9999878258190843
- Test R2: 0.6501665861649835

Model: RF

- Train MAE: 6838.962943084159
- Test MAE: 17561.989321266967
- Train RMSE: 9737.76433682984
- Test RMSE: 24909.3850062178
- Train R2: 0.968826636808535
- Test R2: 0.8213111596362501

Model: XGB

- Train MAE: 12128.153580803064
- Test MAE: 16813.20620050905
- Train RMSE: 16614.262790629655

- Test RMSE: 23598.212733249646
- Train R2: 0.9092540418801894
- Test R2: 0.8396275944222351

and XGBoost has the highest values in test and train set so we've picked XGBoost model

Train R2: 0.9092540418801894 Test R2: 0.8396275944222351

```
# from sklearn.linear_model import LinearRegression
# from sklearn.svm import SVR
# from sklearn.neighbors import KNeighborsRegressor
# from sklearn.tree import DecisionTreeRegressor
# from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score


models = {

    # 'LR': LinearRegression(),
    # 'SVM': SVR(),
    # 'KNN': KNeighborsRegressor(),
    # 'DT': DecisionTreeRegressor(),
    # 'RF': RandomForestRegressor(),
    'XGB': XGBRegressor()
}


for name, model in models.items():
    print(f"Model: {name}")
    print("-"*20)
    model.fit(x_train, y_train)
    y_pred_train = model.predict(x_train)
    y_pred_test = model.predict(x_test)

    print(f"Train MAE: {mean_absolute_error(y_train, y_pred_train)}")
    print(f"Test MAE: {mean_absolute_error(y_test, y_pred_test)}")
    print(f"Train RMSE: {np.sqrt(mean_squared_error(y_train, y_pred_train))}")
```

```
    print(f"Test RMSE: {np.sqrt(mean_squared_error(y_test, y_pred_test))}")
    print(f"Train R2: {r2_score(y_train, y_pred_train)}")
    print(f"Test R2: {r2_score(y_test, y_pred_test)}")
    print("\n")
```

```
Model: XGB
--------------------
[00:40:19] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:square
Train MAE: 12128.153580803064
Test MAE: 16813.20620050905
Train RMSE: 16614.262790629655
Test RMSE: 23598.212733249646
Train R2: 0.9092540418801894
Test R2: 0.8396275944222351
```

```
import joblib

inputs = X.columns

joblib.dump(model, 'model.h5')
joblib.dump(scaler, 'scaler.h5')
joblib.dump(inputs, 'input.h5')
```

```
['input.h5']
```