

IMPORTS

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
```

READING DATA FROM FILE

```
df = pd.read_csv('/kaggle/input/home-data-for-ml-course/train.csv')
```

FILLING IN THE GAPS

```
df[df.select_dtypes(include=['int', 'float']).columns] = df.select_dtypes(include=['int',
df[df.select_dtypes(include=['object']).columns] = df.select_dtypes(include=['object']).f
```

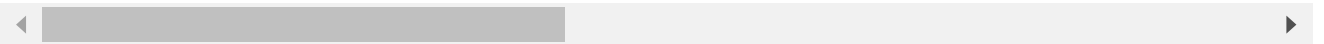
DATA ANALYSIS

```
df.head(3)
```



	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour
0	1	60	RL	65.0	8450	Pave		Reg	L
1	2	20	RL	80.0	9600	Pave		Reg	L
2	3	60	RL	68.0	11250	Pave		IR1	L

3 rows × 81 columns



```
df.info()
```



```

37  BsmtCondSF      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     1460 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    1460 non-null object
58  GarageType     1460 non-null object
59  GarageYrBlt    1460 non-null float64
60  GarageFinish   1460 non-null object
61  GarageCars     1460 non-null int64
62  GarageArea     1460 non-null int64
63  GarageQual     1460 non-null object
64  GarageCond     1460 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         1460 non-null object
73  Fence          1460 non-null object
74  MiscFeature     1460 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

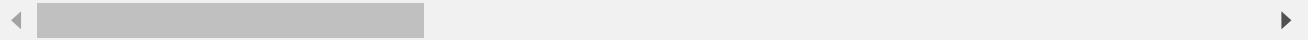
```

```
df.describe()
```



	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000
mean	730.500000	56.897260	57.623288	10516.828082	6.099315	5.575342
std	421.610009	42.300571	34.664304	9981.264932	1.382997	1.112799
min	1.000000	20.000000	0.000000	1300.000000	1.000000	1.000000
25%	365.750000	20.000000	42.000000	7553.500000	5.000000	5.000000
50%	730.500000	50.000000	63.000000	9478.500000	6.000000	5.000000
75%	1095.250000	70.000000	79.000000	11601.500000	7.000000	6.000000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000

8 rows × 38 columns



```
# Finding int and float type columns
```

```
obj_columns = df.select_dtypes(include=['object']).columns.tolist()
```

```
int_columns = df.select_dtypes(include=['int']).columns.tolist() # Find integer column
```

```
float_columns = df.select_dtypes(include=['float']).columns.tolist() # Find float column
```

```
# Printing the results
```

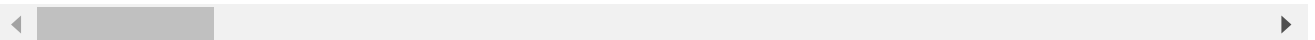
```
print("Object Columns:", obj_columns) # Print object columns
```

```
print("Integer Columns:", int_columns) # Print integer columns
```

```
print("Float Columns:", float_columns) # Print float columns
```



```
Object Columns: ['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities',
Integer Columns: ['Id', 'MSSubClass', 'LotArea', 'OverallQual', 'OverallCond', 'YearB
Float Columns: ['LotFrontage', 'MasVnrArea', 'GarageYrBlt']
```



CONVERTING DATA TO NUMERIC

```
le = LabelEncoder()
```

```
for col in obj_columns:
```

```
    df[col] = le.fit_transform(df[col])
```

SEPARATING DATA

```
# Finding columns where more than 75% of the values are zero
```

```
threshold = 0.75 # 75% threshold
```

```
unimp_num_features = [col for col in df.columns if (df[col] == 0).sum() / len(df) > thres
```

```
# Printing the results
```

```
print("Unimportant numerical features (unimp_num_features):", len(unimp_num_features), ":
```

Unimportant numerical features (unimp_num_features): 15 : ['Alley', 'Utilities', 'Lan

PREPARING THE TRAINING DATA

```
y = df['SalePrice']  
X = df.drop(['Id', 'SalePrice']+unimp_num_features, axis=1)
```

```
X_train, x_test, y_train, y_test = train_test_split(X, y, random_state=60, train_size=0.9)
```

```
X.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	LotConf
0	60	3	65.0	8450	1	3	3	
1	20	3	80.0	9600	1	3	3	
2	60	3	68.0	11250	1	0	3	
3	70	3	60.0	9550	1	0	3	
4	60	3	84.0	14260	1	0	3	

5 rows × 64 columns

```
X.info()
```

8	Neighborhood	1460	non-null	int64
9	Condition1	1460	non-null	int64
10	Condition2	1460	non-null	int64
11	HouseStyle	1460	non-null	int64
12	OverallQual	1460	non-null	int64
13	OverallCond	1460	non-null	int64
14	YearBuilt	1460	non-null	int64
15	YearRemodAdd	1460	non-null	int64
16	RoofStyle	1460	non-null	int64
17	RoofMatl	1460	non-null	int64
18	Exterior1st	1460	non-null	int64
19	Exterior2nd	1460	non-null	int64

```

33 Heating          1460 non-null int64
34 HeatingQC        1460 non-null int64
35 CentralAir       1460 non-null int64
36 Electrical       1460 non-null int64
37 1stFlrSF          1460 non-null int64
38 2ndFlrSF          1460 non-null int64
39 GrLivArea         1460 non-null int64
40 BsmtFullBath      1460 non-null int64
41 FullBath          1460 non-null int64
42 HalfBath          1460 non-null int64
43 BedroomAbvGr     1460 non-null int64
44 KitchenAbvGr     1460 non-null int64
45 KitchenQual       1460 non-null int64
46 TotRmsAbvGrd     1460 non-null int64
47 Functional        1460 non-null int64
48 Fireplaces        1460 non-null int64
49 FireplaceQu       1460 non-null int64
50 GarageType        1460 non-null int64
51 GarageYrBlt       1460 non-null float64
52 GarageFinish      1460 non-null int64
53 GarageCars        1460 non-null int64
54 GarageArea        1460 non-null int64
55 GarageQual        1460 non-null int64
56 GarageCond        1460 non-null int64
57 PavedDrive        1460 non-null int64
58 WoodDeckSF        1460 non-null int64
59 OpenPorchSF       1460 non-null int64
60 MoSold            1460 non-null int64
61 YrSold            1460 non-null int64
62 SaleType          1460 non-null int64
63 SaleCondition     1460 non-null int64
dtypes: float64(3), int64(61)
memory usage: 730.1 KB

```

TRAINING THE MODEL

- LGB Model

```

import lightgbm as lgb
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
# Creating a LightGBM Dataset
train_data = lgb.Dataset(X, label=y)

# Define Model Parameters
params = {
    'objective': 'regression',
    'metric': 'rmse',
    'boosting_type': 'dart',
    'num_leaves': 64,
    'learning_rate': 0.25,
    'feature_fraction': 0.6,
    'verbose': -1
}

# Train the Model
lgb_model = lgb.train(params, train_data, num_boost_round=1000)

```

```
# Make Predictions on the Test Set
y_pred = lgb_model.predict(x_test, num_iteration=lgb_model.best_iteration)
# Evaluate the Performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
```



Mean Squared Error: 5316154.331322902
R² Score: 0.9993859149678057

PROCEDURES REQUIRED FOR TESTING

```
df_test = pd.read_csv('/kaggle/input/home-data-for-ml-course/test.csv.gz')
df_test[df_test.select_dtypes(include=['object']).columns] = df_test.select_dtypes(include=['object']).columns.astype(object)
df_test[df_test.select_dtypes(include=['int', 'float']).columns] = df_test.select_dtypes(include=['int', 'float']).columns.astype(float)
for col in obj_columns:
    df_test[col] = le.fit_transform(df_test[col])
x_test_t2 = df_test.drop(['Id']+unimp_num_features, axis=1)
```

SUMMIT

```
predictions = lgb_model.predict(x_test_t2)

ids = range(1461, 1461 + len(predictions))

submission = pd.DataFrame({'Id': ids, 'SalePrice': predictions})

submission.to_csv('submission.csv', index=False)
```

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.