In [ ]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-pytho
n
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files
under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserve
d as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of
the current session
```

In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:

```
df_train=pd.read_csv("../input/house-prices-advanced-regression-techniques/train.csv")
df_test=pd.read_csv("../input/house-prices-advanced-regression-techniques/test.csv")
```

In [ ]:

```
df_test.describe()
```

In [ ]:

```
df_train.info()
```

In [ ]:

```
sum(df_train.isna().any())
```

In [ ]:

```
#categorical data
df_train["MiscFeature"].value_counts()
```

In [ ]:

```
list_obj_col=df_train.select_dtypes(include='object')
```

In [ ]:

```
list_obj_col
```

In [ ]:

```
list_obj_col=list(df_train.select_dtypes(include='object').columns)
```

In [ ]:

```
list_obj_col
```

In [ ]:

```
list_num_col=list(df_train.select_dtypes(exclude='object').columns)
```

In [ ]:

```
list_num_col
```

In [ ]:

```
def fillna_all(df):
    for col in list_obj_col:
        df[col].fillna(value=df[col].mode()[0],inplace=True)
    for col in list_num_col:
        df[col].fillna(value=df[col].mean(),inplace=True)
```

In [ ]:

```
fillna_all(df_train)
```

In [ ]:

```
sum(df_train.isna().any())
```

In [ ]:

```
#feature encoding
for col in list_obj_col:
    print(col,":",df_train[col].unique())
```

In [ ]:

```
temp=df_train['Id']
dummy=pd.get_dummies(df_train[list_obj_col],prefix=list_obj_col)
```

In [ ]:

```
dummy
```

In [ ]:

```
df_train.drop(list_obj_col,axis=1,inplace=True)
```

In [ ]:

```
df_train.shape
```

In [ ]:

```
#concate
df_train_final=pd.concat([df_train,dummy],axis=1)
```

In [ ]:

```
df_train_final.shape
```

In [ ]:

```
df_train_final.info()
```

In [ ]:

```
#working on test data
list_num_col.remove('SalePrice')
```

In [ ]:

```
fillna_all(df_test)
```

In [ ]:

```
df_test.info()
```

In [ ]:

```
sum(df_test.isna().any())
```

In [ ]:

```
#encoding
dummy1=pd.get_dummies(df_test[list_obj_col],prefix=list_obj_col)
```

In [ ]:

```
dummy1.shape
```

In [ ]:

```
dummy.shape
```

In [ ]:

```
#concatenating
df_train=pd.read_csv("../input/house-prices-advanced-regression-techniques/train.csv")
df_test=pd.read_csv("../input/house-prices-advanced-regression-techniques/test.csv")
```

In [ ]:

```
list_num_col+list_obj_col
```

In [ ]:

```
df_train_test=pd.concat([df_train.drop('SalePrice',axis=1),df_test],axis=0)
```

In [ ]:

```
df_train_test.shape
```

In [ ]:

```
fillna_all(df_train_test)
```

In [ ]:

```
df_train_test.info()
```

In [ ]:

```
dummy2=pd.get_dummies(df_train_test[list_obj_col],prefix=list_obj_col)
```

In [ ]:

```
dummy2.shape
```

In [ ]:

```
df_train_test.drop(list_obj_col,axis=1,inplace=True)
```

In [ ]:

```
df_train_test.shape
```

In [ ]:

```
df_train_test_final=pd.concat([df_train_test,dummy2],axis=1)
```

In [ ]:

```
df_train_test_final.shape
```

In [ ]:

```
df_train_test_final.head()
```

In [ ]:

```
X_train=df_train_test_final.iloc[0:1460]
X_test=df_train_test_final.iloc[1460:]
```

In [ ]:

```
X_train.shape,X_test.shape
```

In [ ]:

```
y=df_train['SalePrice']
```

In [ ]:

```
from sklearn.ensemble import RandomForestRegressor
model=RandomForestRegressor(random_state=23)
```

In [ ]:

```
model.fit(X_train,y)
```

In [ ]:

```
y_predict=model.predict(X_test)
```

In [ ]:

```
y_predict
```

In [ ]:

```
output = pd.DataFrame({'Id': df_test.Id,
                       'SalePrice': y_predict})
output.to_csv('submission.csv', index=False)
```