```
# HOUSE PRICE PREDICTION CODE


import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv("HousePricePrediction.csv")

# numeric columns
numeric_cols = data.select_dtypes(include=['int64', 'float64']).columns

# categorical columns
categorical_cols = data.select_dtypes(include=['object']).columns

Index(['Id', 'MSSubClass', 'LotArea', 'OverallCond', 'YearBuilt',
       'YearRemodAdd', 'BsmtFinSF2', 'TotalBsmtSF', 'SalePrice'],
      dtype='object')

plt.figure(figsize=(12, 6))
sns.heatmap(data.corr(),
                      cmap = 'BrBG',
                      fmt = '.2f',
                      linewidths = 2,
                      annot = True)
plt.title('Heatmap of Numerical Features')


unique_values = []
for col in categorical_cols:
  unique_values.append(data[col].unique().size)
plt.figure(figsize=(10,6))
plt.title('No. Unique values of Categorical Features')
sns.barplot(x=categorical_cols,y=unique_values, palette='colorblind')



plt.figure(figsize=(18, 36))
plt.title('Categorical Features: Distribution')
plt.xticks(rotation=90)
index = 1

for col in categorical_cols:
       y = data[col].value_counts()
       plt.subplot(11, 2, index)
       plt.xticks(rotation=90)
       sns.barplot(x=list(y.index), y=y, palette='muted')
       index += 1


data.drop(['Id'], axis=1, inplace=True)


data.head(5)

data.isnull().sum()


data['MSZoning'] = data['MSZoning'].fillna('NA')
data['Exterior1st'] = data['Exterior1st'].fillna('NA')
```

```
data['SalePrice'] = data['SalePrice'].fillna(data['SalePrice'].mean())
data['BsmtFinSF2'] = data['BsmtFinSF2'].fillna(data['BsmtFinSF2'].mean())
data['TotalBsmtSF'] = data['TotalBsmtSF'].fillna(data['TotalBsmtSF'].mean())


data.isnull().sum()


from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()

data['MSZoning'] = encoder.fit_transform(data['MSZoning'])
data['LotConfig'] = encoder.fit_transform(data['LotConfig'])
data['BldgType'] = encoder.fit_transform(data['BldgType'])
data['Exterior1st'] = encoder.fit_transform(data['Exterior1st'])


from sklearn.model_selection import train_test_split

x = data.drop(['SalePrice'], axis=1)
y = data['SalePrice']

xtrain, xtest, ytrain, ytest = train_test_split(
       x, y, train_size=0.8, test_size=0.2, random_state=0)


from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_percentage_error

rf = RandomForestRegressor(n_estimators=100, random_state=0)
rf.fit(xtrain, ytrain)
Y_pred = rf.predict(xtest)


mape_rf = (mean_absolute_percentage_error(ytest, Y_pred))
mape_rf


from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_percentage_error

dt = DecisionTreeRegressor()
dt.fit(xtrain, ytrain)
Y_pred = dt.predict(xtest)

mape_dt = (mean_absolute_percentage_error(ytest, Y_pred))
mape_dt


from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_percentage_error

gb = GradientBoostingRegressor()
gb.fit(xtrain, ytrain)
Y_pred = gb.predict(xtest)

mape_gb = (mean_absolute_percentage_error(ytest, Y_pred))
mape_gb
```

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_absolute_percentage_error

knn = KNeighborsRegressor()
knn.fit(xtrain, ytrain)
Y_pred = knn.predict(xtest)

mape_knn = (mean_absolute_percentage_error(ytest, Y_pred))
mape_knn


from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_percentage_error

lr = LinearRegression()
lr.fit(xtrain, ytrain)
Y_pred = lr.predict(xtest)

mape_lr = (mean_absolute_percentage_error(ytest, Y_pred))
mape_lr


from sklearn import svm
from sklearn.svm import SVC
from sklearn.metrics import mean_absolute_percentage_error

model_SVR = svm.SVR()
model_SVR.fit(xtrain,ytrain)
Y_pred = model_SVR.predict(xtest)

mape_svr=(mean_absolute_percentage_error(ytest, Y_pred))
mape_svr


import matplotlib.pyplot as plt

# Assuming mape_values is a list of your MAPE values
mape_values = [mape_svr, mape_lr, mape_knn, mape_gb, mape_dt, mape_rf]

# Model names
models = ['SVR', 'Linear Regression', 'KNN', 'Gradient Boosting', 'Decision Tree', 'Random
Forest']
color=['skyblue']
plt.figure(figsize=(10, 5))
plt.bar(models, mape_values, color=color)
plt.xlabel('Models')
plt.ylabel('Mean Absolute Percentage Error')
plt.title('Model Performance')
plt.show()


import matplotlib.pyplot as plt
import numpy as np

# Dictionary of models
models = {
    'Random Forest': rf,
    'Decision Tree': dt,
```

```
    'Gradient Boosting': gb,
    'KNN': knn,
}

# Dictionary of MAPE values
models_mape = {
    'Random Forest': mape_rf,
    'Decision Tree': mape_dt,
    'Gradient Boosting': mape_gb,
    'KNN': mape_knn,
}

# Create a line plot for each model
for model_name in models.keys():
    # Predict values
    model = models[model_name]
    Y_pred = model.predict(xtest)

    # Create a line plot
    plt.figure(figsize=(10, 5))
    plt.plot(ytest.values, label='Actual', color='blue')
    plt.plot(Y_pred, label='Predicted', color='red')
    plt.title(f'Actual vs Predicted House Prices ({model_name})')
    plt.xlabel('Number of Houses')
    plt.ylabel('House Price')
    plt.legend()
    plt.show()

colors = ['skyblue', 'red', 'purple', 'orange', 'yellow']

# Create a bar plot to compare MAPE values
plt.figure(figsize=(10, 5))
plt.bar(models_mape.keys(), models_mape.values(), color=colors)
plt.title('Model Performance Comparison')
plt.xlabel('Model')
plt.ylabel('MAPE')
plt.show()


import seaborn as sns

# Create a bar plot to compare MAPE values
plt.figure(figsize=(10, 5))

# Get the 'colorblind' color palette from seaborn
colors = sns.color_palette('colorblind', len(models_mape))

plt.bar(models_mape.keys(), models_mape.values(), color=colors)
plt.title('Model Performance Comparison')
plt.xlabel('Model')
plt.ylabel('MAPE')
plt.show()
```