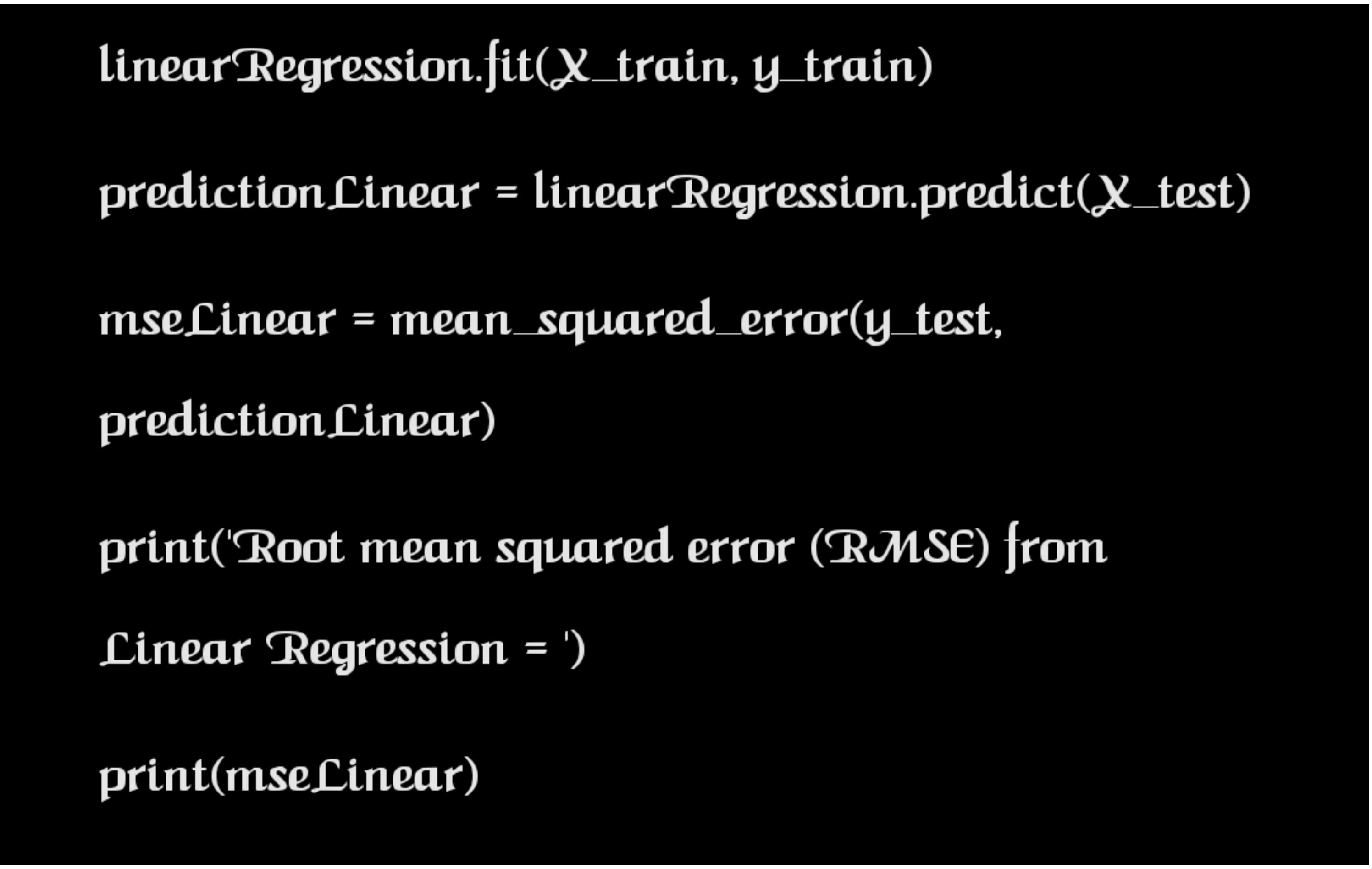
```
from matplotlib.ticker import decade_down

from sklearn import *

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.preprocessing import
LabelEncoder,StandardScaler

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.metrics import mean_squared_error

from sklearn.ensemble import
RandomForestRegressor

from sklearn.datasets import
fetch_california_housing

db = fetch_california_housing(as_frame=True)

%matplotlib inline



import warnings

warnings.filterwarnings('ignore')

print(db.data.head())

print(db.data.describe)
print(db.frame.info())

print(db.frame.head())

print(db.data.columns)

print(db.data.isnull().sum())

names = db.data.columns

# Create the Scaler object

scaler = StandardScaler()

# Fit your data on the scaler object

scaled_db1= scaler.fit_transform(db.data)

scaled_db =

pd.DataFrame(scaled_db1,columns=names)

scaled_db.head()

rng = np.random.RandomState(0)

indices = rng.choice(np.arange(db.frame.shape[0]),

size=500,

                    replace=False)

sns.scatterplot(data=db.frame.iloc[indices],

                x="Longitude", y="Latitude",

          size="MedHouseVal", hue="MedHouseVal",

            palette="viridis", alpha=0.5)

plt.legend(title="MedHouseVal",
bbox_to_anchor=(1.05, 1),

              loc="upper left")

_ = plt.title("Median house value depending of\n
their spatial location")


X = db.data.iloc[:, :-1].values

y = db.data.iloc[:, [-1]].values




X_labelencoder = LabelEncoder()

X[:, -1] = X_labelencoder.fit_transform(X[:, -1])

X_train, X_test, y_train, y_test = train_test_split(X,
y,test_size = 0.2,

                                             random_state = 0)


scaler1 = StandardScaler()

X_train = scaler1.fit_transform(X_train)

X_test = scaler1.transform(X_test)

y_train = scaler1.fit_transform(y_train)


y_test = scaler1.transform(y_test)

linearRegression = LinearRegression()
```



```
linearRegression.fit(X_train, y_train)

predictionLinear = linearRegression.predict(X_test)

mseLinear = mean_squared_error(y_test,
predictionLinear)

print('Root mean squared error (RMSE) from
Linear Regression = ')

print(mseLinear)
```