In [196...

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

In [197...

```
housing=pd.read_csv(r"D:\PC\Data Science\data sets\Housing\housing.csv")
```

In [198...

```
housing.head()
```

Out[198...

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0

In [199...

```
housing.columns
```

Out[199...

```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'ocean_proximity'],
      dtype='object')
```

In [200...

```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

In [201...

```
housing["ocean_proximity"].value_counts()
```

Out[201...

```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

In [202...

```
housing.describe()
```

Out[202...

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populatio
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.00000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.47674

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	populatior
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000

In [203…

```
from sklearn.model_selection import train_test_split
```

In [204…

```
train_set, test_set= train_test_split(housing, test_size=0.2, random_state=42)
```

In [205…

```
housing["income_cat"]= pd.cut(housing["median_income"],
                              bins=[0., 1.5, 3.0, 4.5, 6., np.inf],
                              labels=[1, 2, 3, 4, 5])
```

In [206…

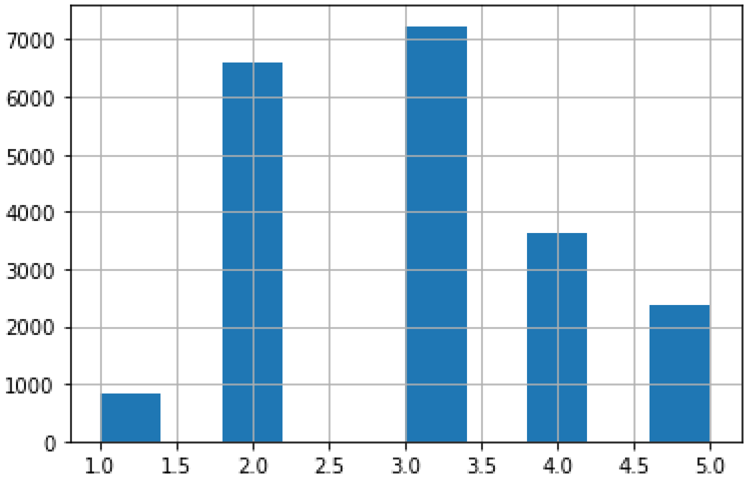
```
housing_labels = train_set["median_house_value"].copy()
```

In [207…

```
housing["income_cat"].hist()
```

Out[207…

```
<AxesSubplot:>
```

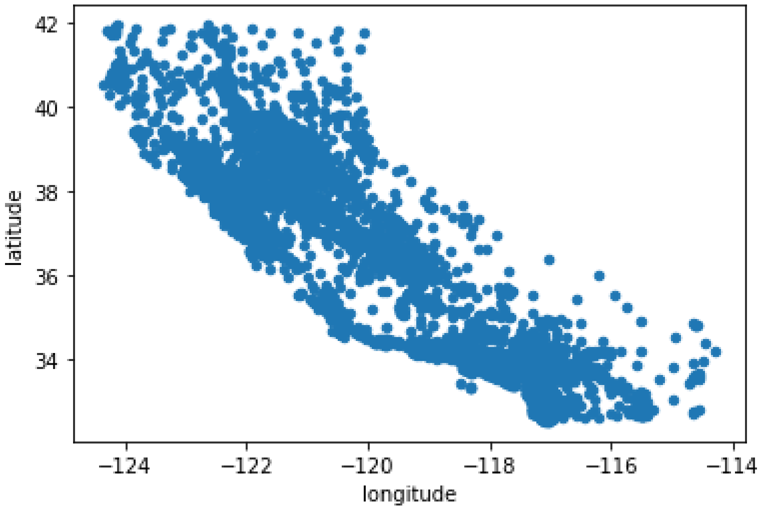


In [208…

```
housing.plot(kind="scatter", x="longitude", y="latitude")
```

Out[208…

```
<AxesSubplot:xlabel='longitude', ylabel='latitude'>
```

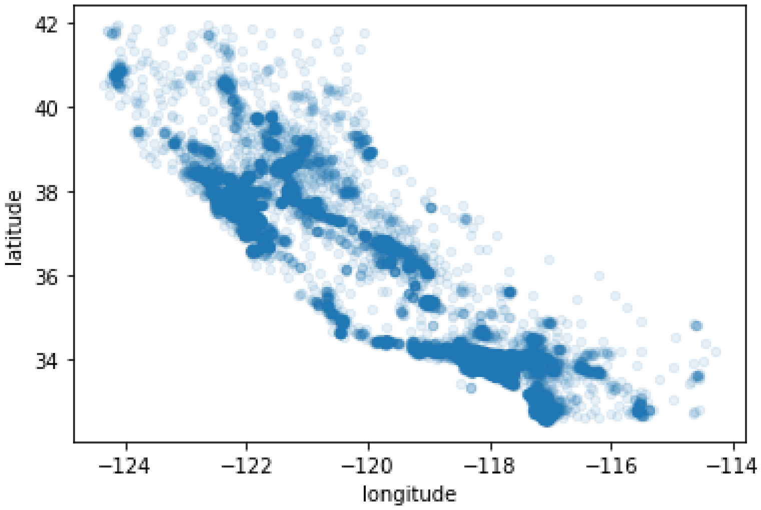


In [209…

```
housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.1)
```

Out[209…

```
<AxesSubplot:xlabel='longitude', ylabel='latitude'>
```

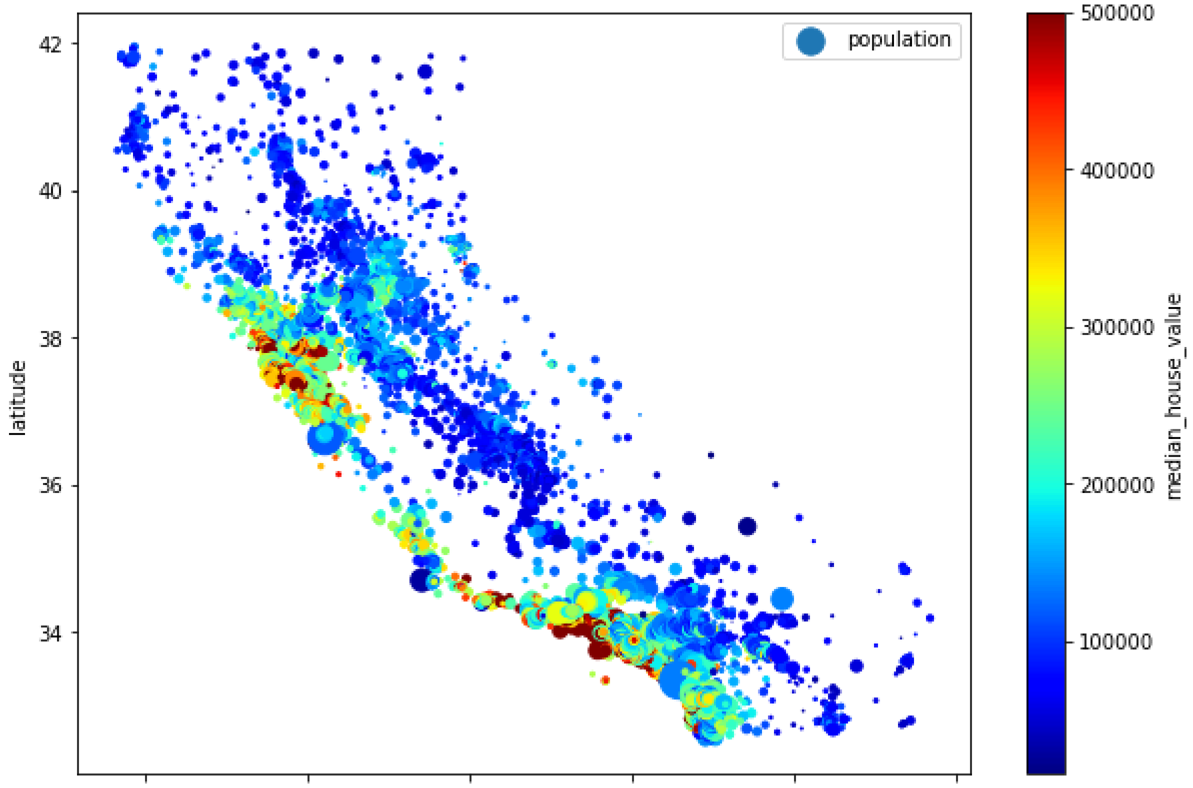


In [210…

```
housing.plot(kind="scatter", x="longitude", y="latitude",
            s=housing["population"]/100, label="population", figsize=(10,7),
            c="median_house_value", cmap=plt.get_cmap("jet"), colorbar=True,)
plt.legend()
```

Out[210…

```
<matplotlib.legend.Legend at 0x2059fe222e0>
```



```
In [211…  corr_matrix= housing.corr()
```

```
In [212…  corr_matrix["median_house_value"].sort_values(ascending=False)
```

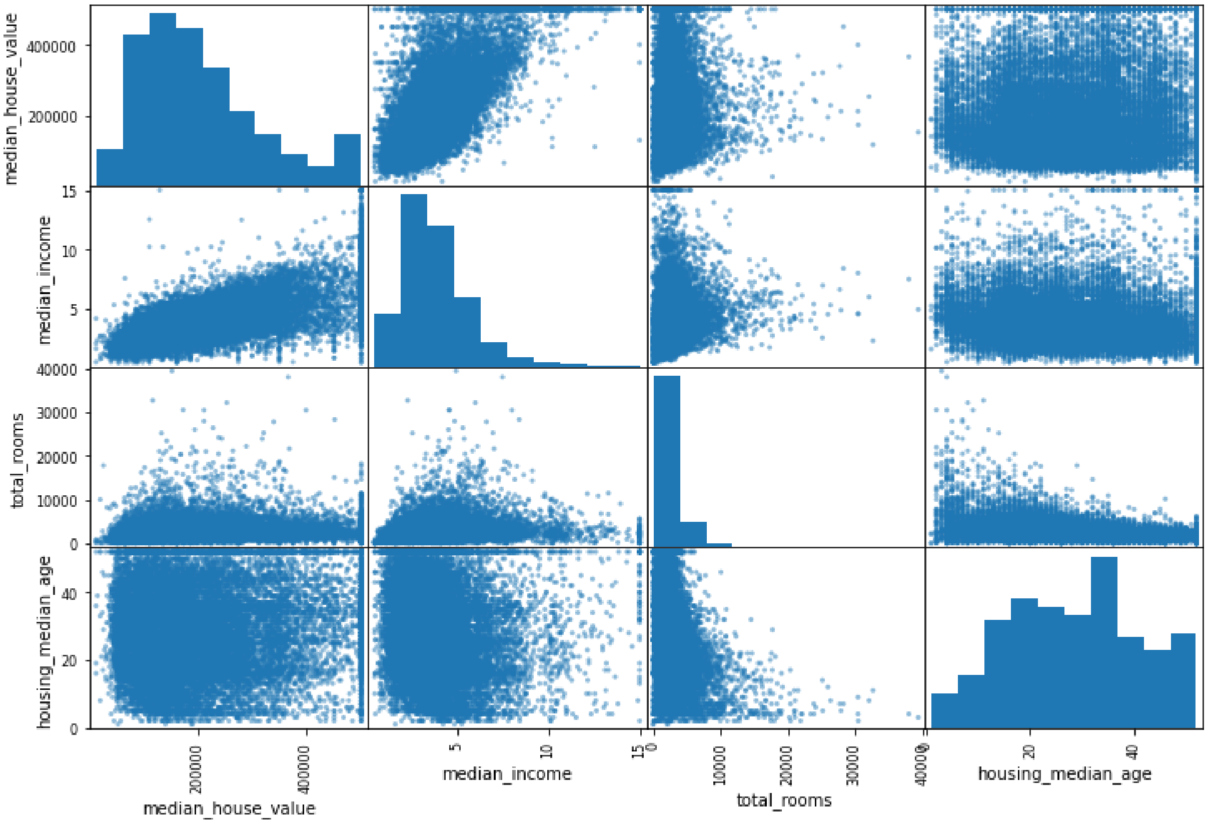
```
Out[212…  median_house_value    1.000000
          median_income         0.688075
          total_rooms           0.134153
          housing_median_age    0.105623
          households            0.065843
          total_bedrooms        0.049686
          population           -0.024650
          longitude            -0.045967
          latitude             -0.144160
          Name: median_house_value, dtype: float64
```

```
In [213…  from pandas.plotting import scatter_matrix
```

```
In [214…  attributes = ["median_house_value", "median_income", "total_rooms", "housing_median_
```

```
In [215…  scatter_matrix(housing[attributes], figsize=(12,8))
```

```
Out[215…  array([[<AxesSubplot:xlabel='median_house_value', ylabel='median_house_value'>,
                  <AxesSubplot:xlabel='median_income', ylabel='median_house_value'>,
                  <AxesSubplot:xlabel='total_rooms', ylabel='median_house_value'>,
                  <AxesSubplot:xlabel='housing_median_age', ylabel='median_house_value'>],
                 [<AxesSubplot:xlabel='median_house_value', ylabel='median_income'>,
                  <AxesSubplot:xlabel='median_income', ylabel='median_income'>,
                  <AxesSubplot:xlabel='total_rooms', ylabel='median_income'>,
                  <AxesSubplot:xlabel='housing_median_age', ylabel='median_income'>],
                 [<AxesSubplot:xlabel='median_house_value', ylabel='total_rooms'>,
                  <AxesSubplot:xlabel='median_income', ylabel='total_rooms'>,
                  <AxesSubplot:xlabel='total_rooms', ylabel='total_rooms'>,
                  <AxesSubplot:xlabel='housing_median_age', ylabel='total_rooms'>],
                 [<AxesSubplot:xlabel='median_house_value', ylabel='housing_median_age'>,
                  <AxesSubplot:xlabel='median_income', ylabel='housing_median_age'>,
                  <AxesSubplot:xlabel='total_rooms', ylabel='housing_median_age'>,
                  <AxesSubplot:xlabel='housing_median_age', ylabel='housing_median_age'>]],
                dtype=object)
```



In [216…

```
median= train_set["total_bedrooms"].median()
```

In [217…

```
train_set["total_bedrooms"].fillna(median, inplace=True)
```

```
<ipython-input-217-9ba39919f72d>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  train_set["total_bedrooms"].fillna(median, inplace=True)
```

In [248…

```
train_set.columns
```

Out[248…

```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'ocean_proximity'],
      dtype='object')
```

In [249…

```
X_train_set=train_set.drop("median_house_value", axis=1)
```

In [250…

```
from sklearn.impute import SimpleImputer
```

In [251…

```
housing_num=X_train_set.drop("ocean_proximity", axis=1)
```

In [252…

```
imputer=SimpleImputer(strategy="median")
```

In [253…

```
imputer.fit(housing_num)
```

Out[253…

```
SimpleImputer(strategy='median')
```

In [254…

```
imputer.statistics_
```

Out[254…

```
array([-118.51  ,   34.26  ,   29.    , 2129.    ,  437.    , 1167.    ,
        410.    ,    3.5458])
```

In [255…
```
housing_num.median().values
```

Out[255…
```
array([-118.51  ,   34.26  ,   29.    , 2129.    ,  437.    , 1167.    ,
        410.    ,    3.5458])
```

In [256…
```
X=imputer.transform(housing_num)
```

In [257…
```
housing_tr=pd.DataFrame(X, columns=housing_num.columns)
```

In [258…
```
housing_cat=train_set[["ocean_proximity"]]
```

In [259…
```
housing_cat.head()
```

Out[259…

	ocean_proximity
14196	NEAR OCEAN
8267	NEAR OCEAN
17445	NEAR OCEAN
14265	NEAR OCEAN
2271	INLAND

In [260…
```
from sklearn.preprocessing import OrdinalEncoder
```

In [261…
```
ordinal_encoder=OrdinalEncoder()
```

In [262…
```
housing_cat_encoded= ordinal_encoder.fit_transform(housing_cat)
```

In [263…
```
housing_cat_encoded[:10]
```

Out[263…
```
array([[4.],
       [4.],
       [4.],
       [4.],
       [1.],
       [0.],
       [0.],
       [3.],
       [0.],
       [0.]])
```

In [264…
```
ordinal_encoder.categories_
```

Out[264…
```
[array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'],
       dtype=object)]
```

In [265…
```
from sklearn.preprocessing import OneHotEncoder
```

In [266…
```
cat_encoder=OneHotEncoder()
```

In [267…
```
housing_cat_1hot= cat_encoder.fit_transform(housing_cat)
```

In [268…
```
housing_cat_1hot
```

Out[268…
```
<16512x5 sparse matrix of type '<class 'numpy.float64'>'
	with 16512 stored elements in Compressed Sparse Row format>
```

In [269…
```
housing_cat_1hot.toarray()
```

Out[269…

```
array([[0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 1.],
       ...,
       [1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0.]])
```

In [270…

```
cat_encoder.categories_
```

Out[270…

```
[array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'],
       dtype=object)]
```

In [271…

```
from sklearn.base import BaseEstimator, TransformerMixin
```

In [272…

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
```

In [273…

```
num_pipeline= Pipeline([
    ('imputer', SimpleImputer(strategy="median")),
    ('std_scaler', StandardScaler()),
])
```

In [274…

```
housing_num_tr=num_pipeline.fit_transform(housing_num)
```

In [275…

```
from sklearn.compose import ColumnTransformer
```

In [276…

```
num_attribs= list(housing_num)
cat_attribs= ["ocean_proximity"]
```

In [345…

```
full_pipeline= ColumnTransformer([
    ("num", num_pipeline, num_attribs),
    ("cat", OneHotEncoder(), cat_attribs),
])
```

In [351…

```
train_set.head()
```

Out[351…

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househo
14196	-117.03	32.71	33.0	3126.0	627.0	2300.0	62
8267	-118.16	33.77	49.0	3382.0	787.0	1314.0	75
17445	-120.48	34.66	4.0	1897.0	331.0	915.0	33
14265	-117.11	32.69	36.0	1421.0	367.0	1418.0	35
2271	-119.80	36.78	43.0	2382.0	431.0	874.0	38

In [347…

```
housing_prepared= full_pipeline.fit_transform(train_set)
```

In [279…

```
from sklearn.linear_model import LinearRegression
```

In [280…

```
lin_reg= LinearRegression()
```

In [281…

```
lin_reg.fit(housing_prepared, housing_labels)
```

Out[281…

```
LinearRegression()
```

In [282…

```
from sklearn.metrics import mean_squared_error
```

In [283…

```
housing_predictions= lin_reg.predict(housing_prepared)
```

In [284…

```
lin_mse= mean_squared_error(housing_labels, housing_predictions)
```

In [285…

```
lin_rmse= np.sqrt(lin_mse)
```

In [286…

```
lin_rmse
```

Out[286…

```
68433.93736666226
```

In [287…

```
from sklearn.metrics import r2_score
```

In [288…

```
r2_score(housing_labels, housing_predictions)
```

Out[288…

```
0.6496648627123223
```

In [289…

```
from sklearn.tree import DecisionTreeRegressor
```

In [290…

```
tree_reg= DecisionTreeRegressor()
```

In [291…

```
tree_reg.fit(housing_prepared, housing_labels)
```

Out[291…

```
DecisionTreeRegressor()
```

In [292…

```
housing_predictions= tree_reg.predict(housing_prepared)
```

In [293…

```
tree_mse=mean_squared_error(housing_labels, housing_predictions)
```

In [294…

```
tree_rmse=np.sqrt(tree_mse)
```

In [295…

```
tree_rmse
```

Out[295…

```
0.0
```

In [296…

```
from sklearn.model_selection import cross_val_score
```

In [301…

```
scores= cross_val_score(tree_reg, housing_prepared, housing_labels,
                        scoring="neg_mean_squared_error", cv=10)
```

In [302…

```
tree_rmse_scores= np.sqrt(-scores)
```

In [303…

```
def display_scores(scores):
    print("Scores", scores)
    print("Mean", scores.mean())
    print("SD", scores.std())
```

In [304…

```
display_scores(tree_rmse_scores)
```

```
Scores [66276.3541399  67726.07353866 67990.65131197 72371.60417774
 66988.98321269 66475.391629   62779.49161703 69994.77542875
 70474.54007364 67118.74896264]
Mean 67819.66140920234
SD 2516.170812019729
```

In [311…

```
from sklearn.ensemble import RandomForestRegressor
```

In [312…

```
forest_reg= RandomForestRegressor()
```

In [313…

```
forest_reg.fit(housing_prepared, housing_labels)
```

Out[313…

```
RandomForestRegressor()
```

In [314…

```
housing_predictions= forest_reg.predict(housing_prepared)
```

In [315…

```
forest_mse=mean_squared_error(housing_labels, housing_predictions)
```

In [316…

```
forest_rmse=np.sqrt(forest_mse)
```

In [317…

```
forest_rmse
```

Out[317…

```
18094.098051516652
```

In [318…

```
scores= cross_val_score(forest_reg, housing_prepared, housing_labels,
                        scoring="neg_mean_squared_error", cv=10)
```

In [319…

```
forest_rmse_scores= np.sqrt(-scores)
```

In [320…

```
display_scores(forest_rmse_scores)
```

```
Scores [46925.29728707 50633.14879565 47809.58201204 50158.44551746
 49777.14297837 46489.55801507 45669.27129373 50812.68309854
 49381.10350237 49709.55915788]
Mean 48736.579165819196
SD 1759.866755098333
```

In [321…

```
r2_score(housing_labels, housing_predictions)
```

Out[321…

```
0.9755085487322805
```

In [322…

```
from sklearn.model_selection import GridSearchCV
```

In [325…

```
param_grid=[
    {'n_estimators':[3,10,30], 'max_features':[2,4,6,8]},
    {'bootstrap':[False], 'n_estimators':[3,10], 'max_features':[2,3,4]},
]
```

In [326…

```
forest_reg= RandomForestRegressor()
```

In [327…

```
grid_search=GridSearchCV(forest_reg, param_grid, cv=5,
                        scoring='neg_mean_squared_error',
                        return_train_score=True)
```

In [328…

```
grid_search.fit(housing_prepared, housing_labels)
```

Out[328…

```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid=[{'max_features': [2, 4, 6, 8],
                          'n_estimators': [3, 10, 30]},
                         {'bootstrap': [False], 'max_features': [2, 3, 4],
                          'n_estimators': [3, 10]}],
             return_train_score=True, scoring='neg_mean_squared_error')
```

In [329…

```
grid_search.best_params_
```

Out[329…

```
{'max_features': 8, 'n_estimators': 30}
```

In [330…

```
grid_search.best_estimator_
```

Out[330…

```
RandomForestRegressor(max_features=8, n_estimators=30)
```

In [331…

```
cvres= grid_search.cv_results_
```

In [333…

```
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
    print(np.sqrt(-mean_score), params)
```

```
62692.4654345404 {'max_features': 2, 'n_estimators': 3}
55069.50733235221 {'max_features': 2, 'n_estimators': 10}
52110.59077103753 {'max_features': 2, 'n_estimators': 30}
58955.342263822124 {'max_features': 4, 'n_estimators': 3}
52297.74217779409 {'max_features': 4, 'n_estimators': 10}
50181.75344023495 {'max_features': 4, 'n_estimators': 30}
58999.39400349141 {'max_features': 6, 'n_estimators': 3}
52285.04447774024 {'max_features': 6, 'n_estimators': 10}
49858.03618233968 {'max_features': 6, 'n_estimators': 30}
58826.82351294995 {'max_features': 8, 'n_estimators': 3}
51469.744462704686 {'max_features': 8, 'n_estimators': 10}
49541.36348219184 {'max_features': 8, 'n_estimators': 30}
62069.12781237627 {'bootstrap': False, 'max_features': 2, 'n_estimators': 3}
54450.758248770755 {'bootstrap': False, 'max_features': 2, 'n_estimators': 10}
59410.50337615474 {'bootstrap': False, 'max_features': 3, 'n_estimators': 3}
52428.87435495705 {'bootstrap': False, 'max_features': 3, 'n_estimators': 10}
58138.52190844188 {'bootstrap': False, 'max_features': 4, 'n_estimators': 3}
51251.93030127868 {'bootstrap': False, 'max_features': 4, 'n_estimators': 10}
```

In [334…

```
final_model= grid_search.best_estimator_
```

In [360…

```
X_test=test_set.drop("median_house_value", axis=1)
```

In [342…

```
y_test= test_set["median_house_value"].copy()
```

In [354…

```
test_set.head()
```

Out[354…

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househo
20046	-119.01	36.06	25.0	1505.0	NaN	1392.0	35
3024	-119.46	35.14	30.0	2943.0	NaN	1565.0	58
15663	-122.44	37.80	52.0	3830.0	NaN	1310.0	96
20484	-118.72	34.28	17.0	3051.0	NaN	1705.0	49
9814	-121.93	36.62	34.0	2351.0	NaN	1063.0	42

In [361…

```
test_set.columns
```

Out[361…

```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'ocean_proximity'],
      dtype='object')
```

In [363…

```
X_test_prepared= full_pipeline.transform(test_set)
```

In [364…

```
final_predictions= final_model.predict(X_test_prepared)
```

In [365…

```
final_mse= mean_squared_error(y_test, final_predictions)
```

In [366…

```
final_rmse= np.sqrt(final_mse)
```

In [367…

```
final_rmse
```

Out[367…

```
48586.34629402553
```

In [ ]: