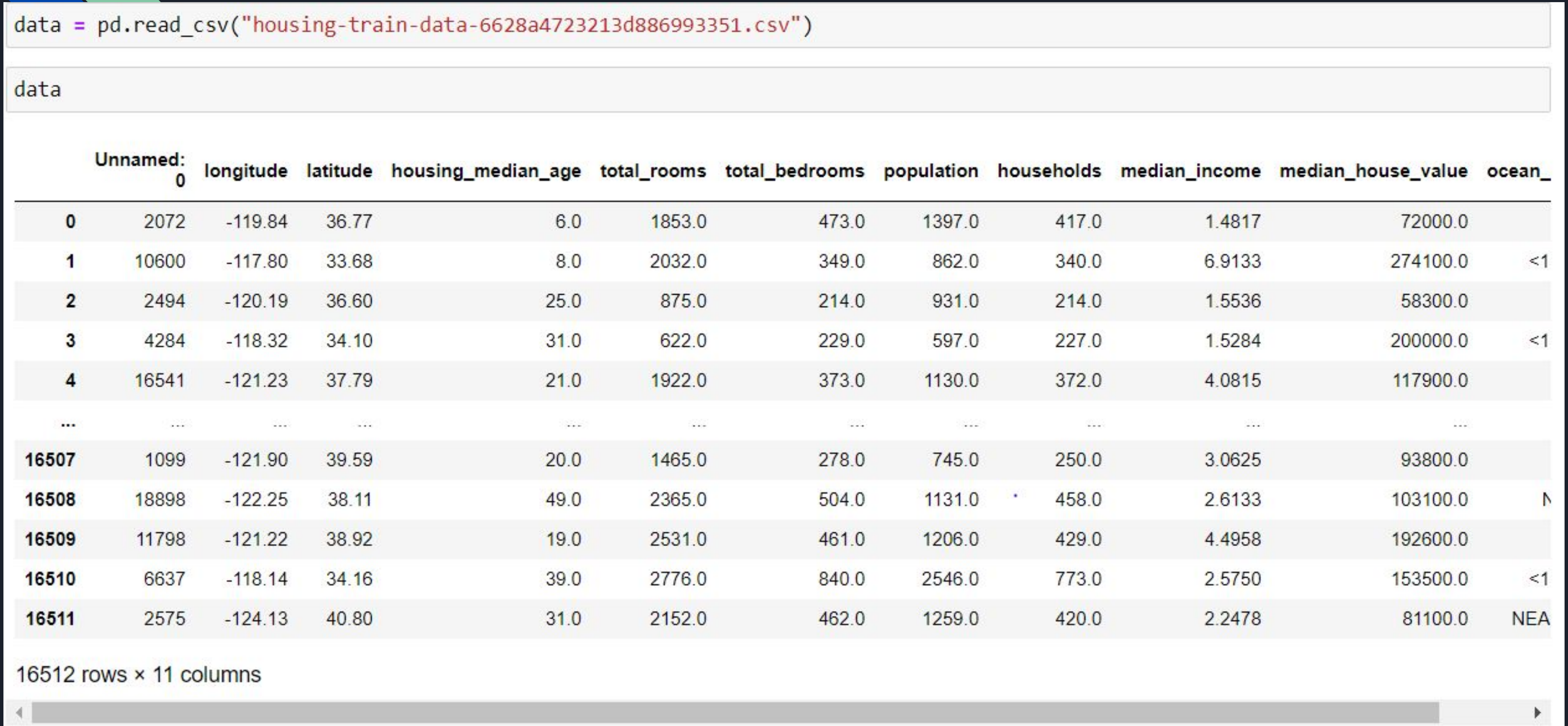
Ahmed et JS immobilier BONJOUR

Notre EDA:

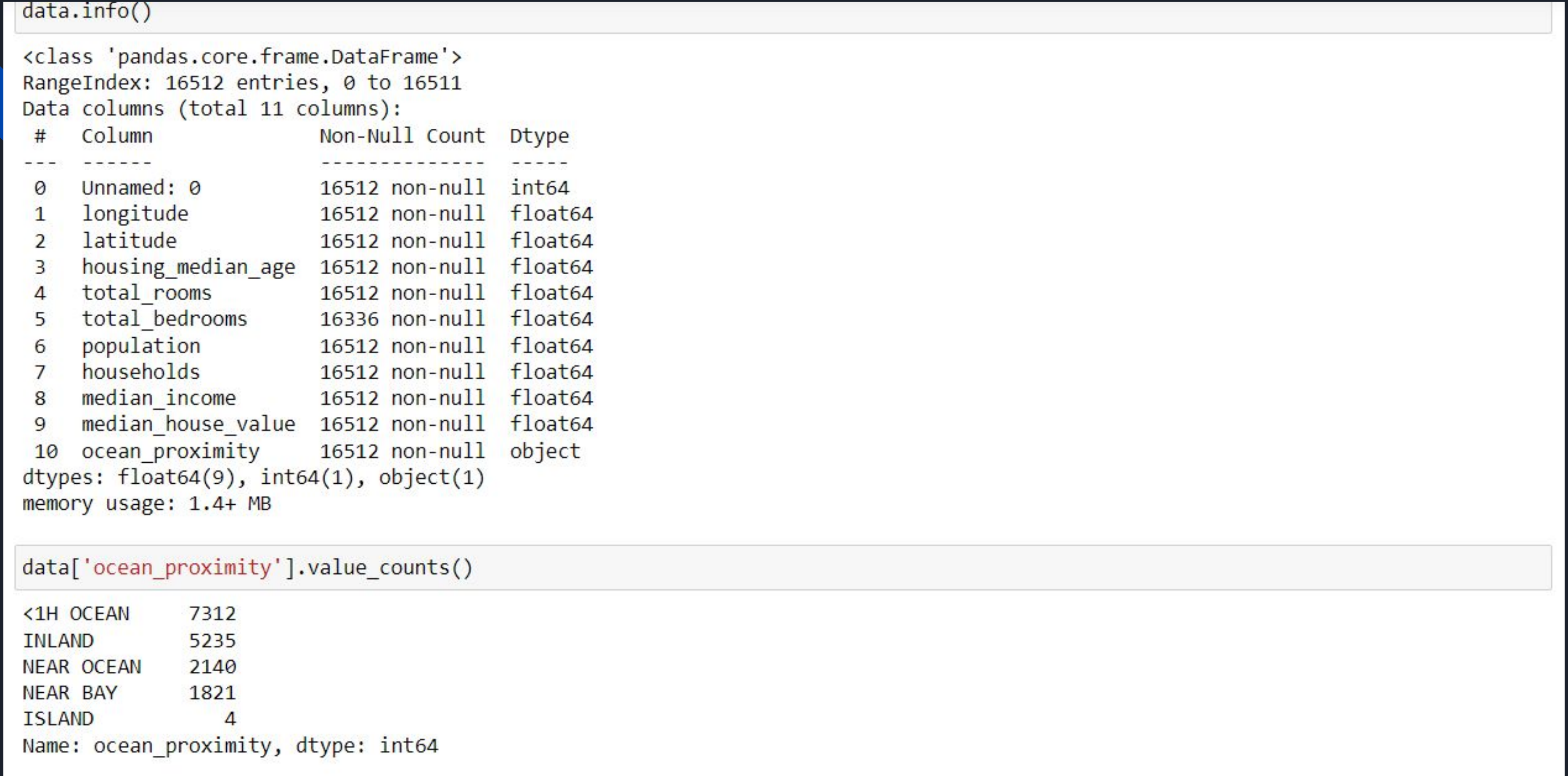


```
data = pd.read_csv("housing-train-data-6628a4723213d886993351.csv")
```

```
data
```

	Unnamed: 0	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_
0	2072	-119.84	36.77	6.0	1853.0	473.0	1397.0	417.0	1.4817	72000.0	
1	10600	-117.80	33.68	8.0	2032.0	349.0	862.0	340.0	6.9133	274100.0	<1
2	2494	-120.19	36.60	25.0	875.0	214.0	931.0	214.0	1.5536	58300.0	
3	4284	-118.32	34.10	31.0	622.0	229.0	597.0	227.0	1.5284	200000.0	<1
4	16541	-121.23	37.79	21.0	1922.0	373.0	1130.0	372.0	4.0815	117900.0	
...	...	...	...	...	...	...	...	...	...	...	
16507	1099	-121.90	39.59	20.0	1465.0	278.0	745.0	250.0	3.0625	93800.0	
16508	18898	-122.25	38.11	49.0	2365.0	504.0	1131.0	458.0	2.6133	103100.0	N
16509	11798	-121.22	38.92	19.0	2531.0	461.0	1206.0	429.0	4.4958	192600.0	
16510	6637	-118.14	34.16	39.0	2776.0	840.0	2546.0	773.0	2.5750	153500.0	<1
16511	2575	-124.13	40.80	31.0	2152.0	462.0	1259.0	420.0	2.2478	81100.0	NEA

16512 rows × 11 columns

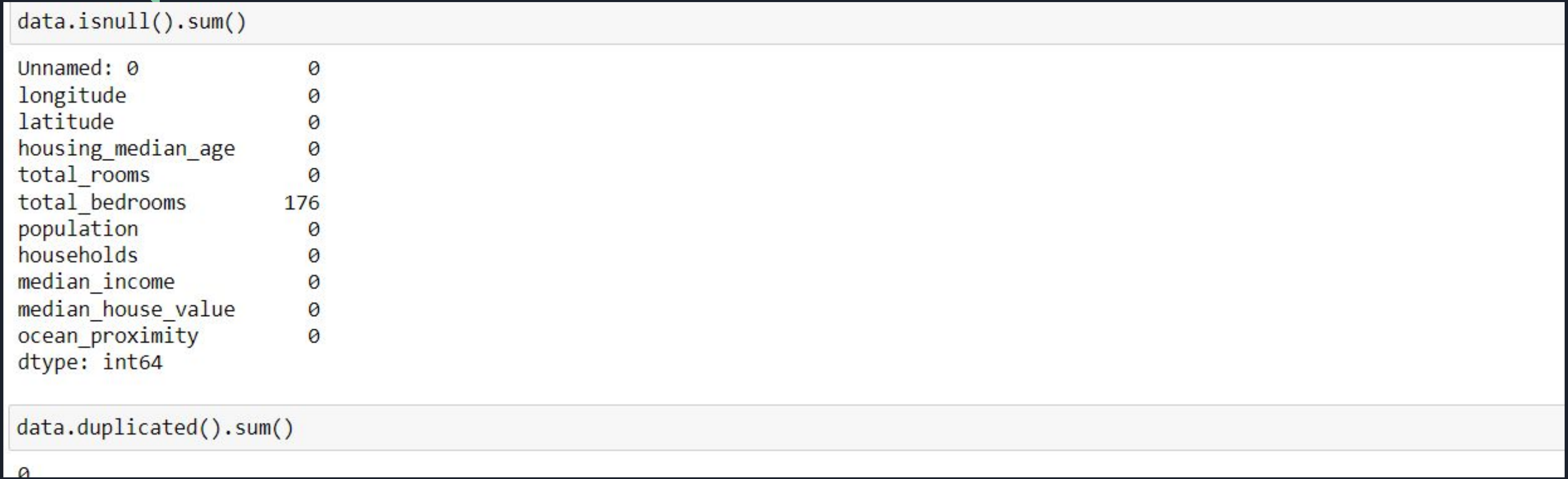


```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16512 entries, 0 to 16511
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Unnamed: 0          16512 non-null  int64  
 1   longitude           16512 non-null  float64
 2   latitude            16512 non-null  float64
 3   housing_median_age  16512 non-null  float64
 4   total_rooms         16512 non-null  float64
 5   total_bedrooms      16336 non-null  float64
 6   population          16512 non-null  float64
 7   households          16512 non-null  float64
 8   median_income       16512 non-null  float64
 9   median_house_value  16512 non-null  float64
 10  ocean_proximity     16512 non-null  object 
dtypes: float64(9), int64(1), object(1)
memory usage: 1.4+ MB
```

```
data['ocean_proximity'].value_counts()
```

```
<1H OCEAN     7312
INLAND        5235
NEAR OCEAN    2140
NEAR BAY      1821
ISLAND           4
Name: ocean_proximity, dtype: int64
```

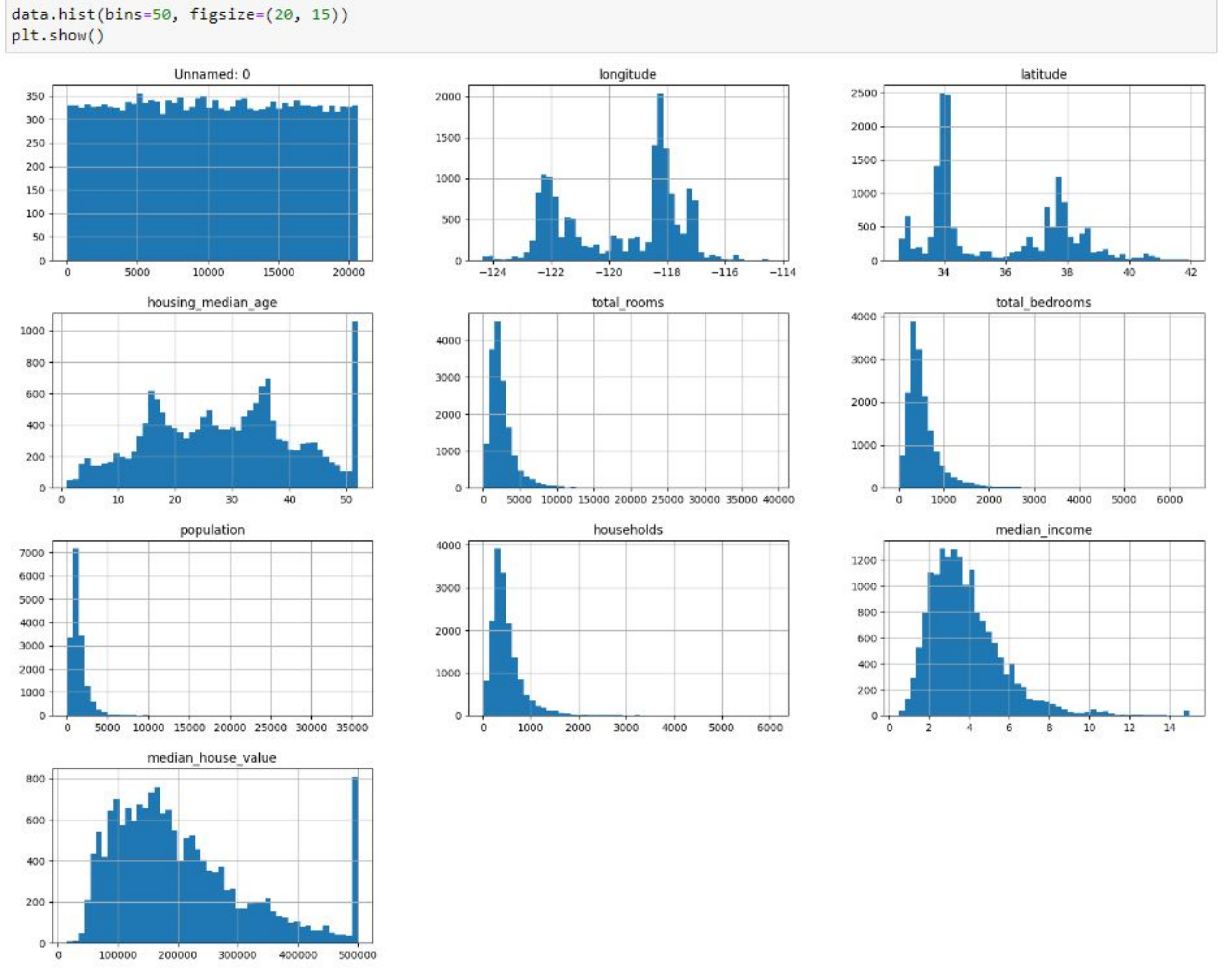


```
data.isnull().sum()
```

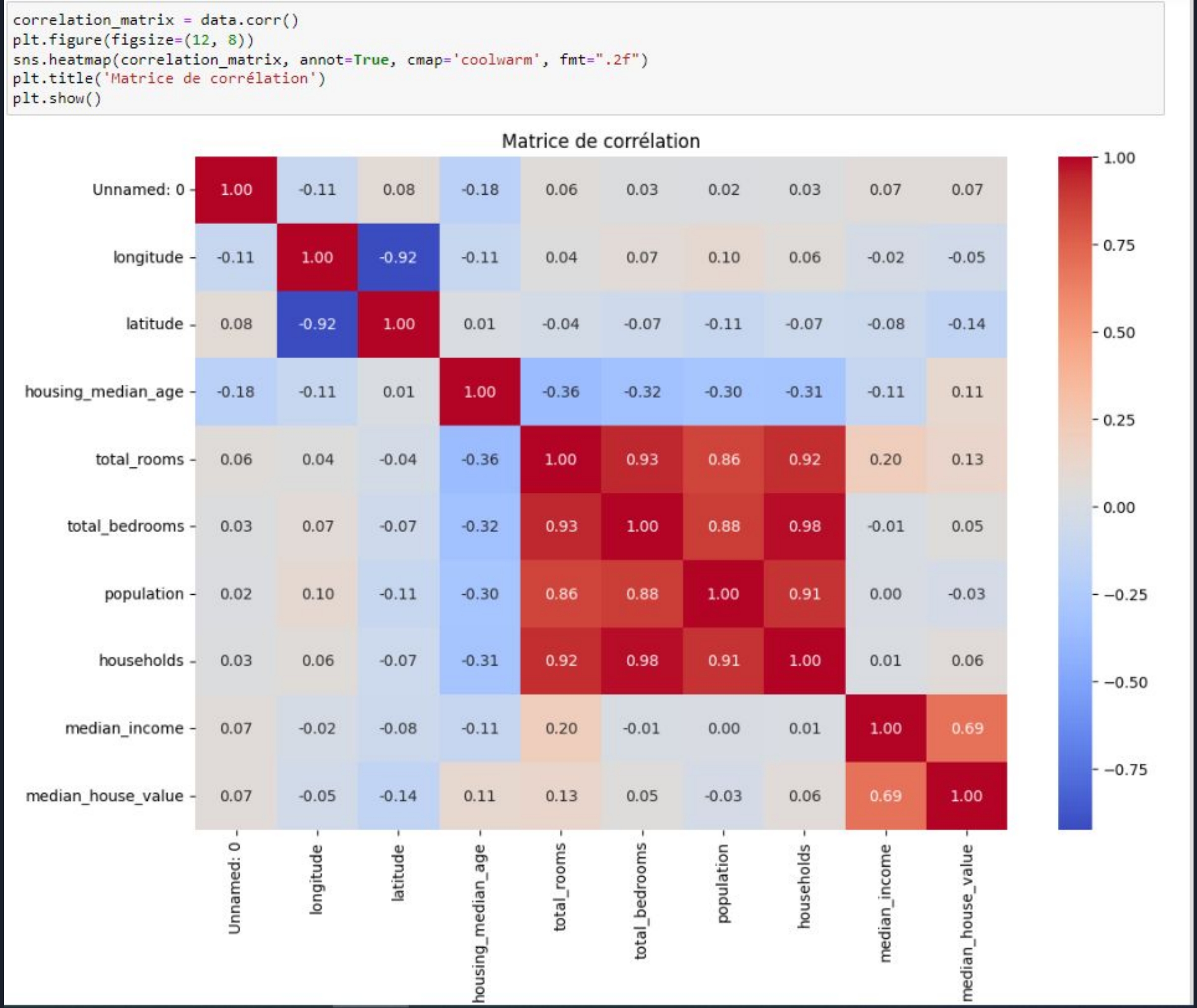
```
Unnamed: 0              0
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        176
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64
```

```
data.duplicated().sum()
```

```
0
```



```
data.hist(bins=50, figsize=(20, 15))
plt.show()
```

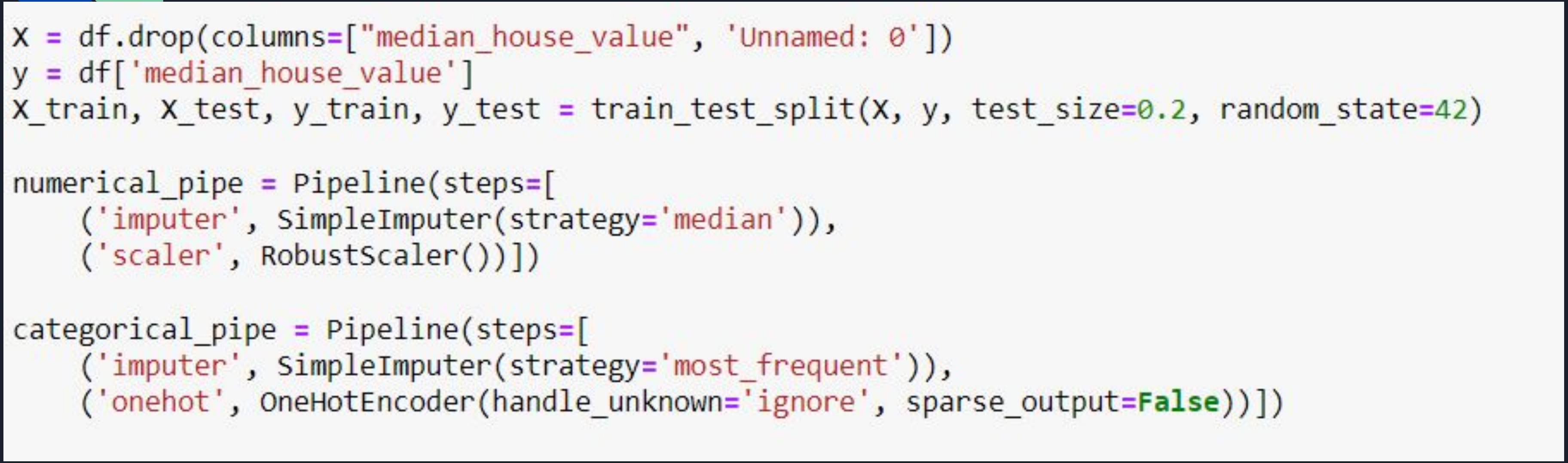


```
correlation_matrix = data.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Matrice de corrélation')
plt.show()
```

Matrice de corrélation

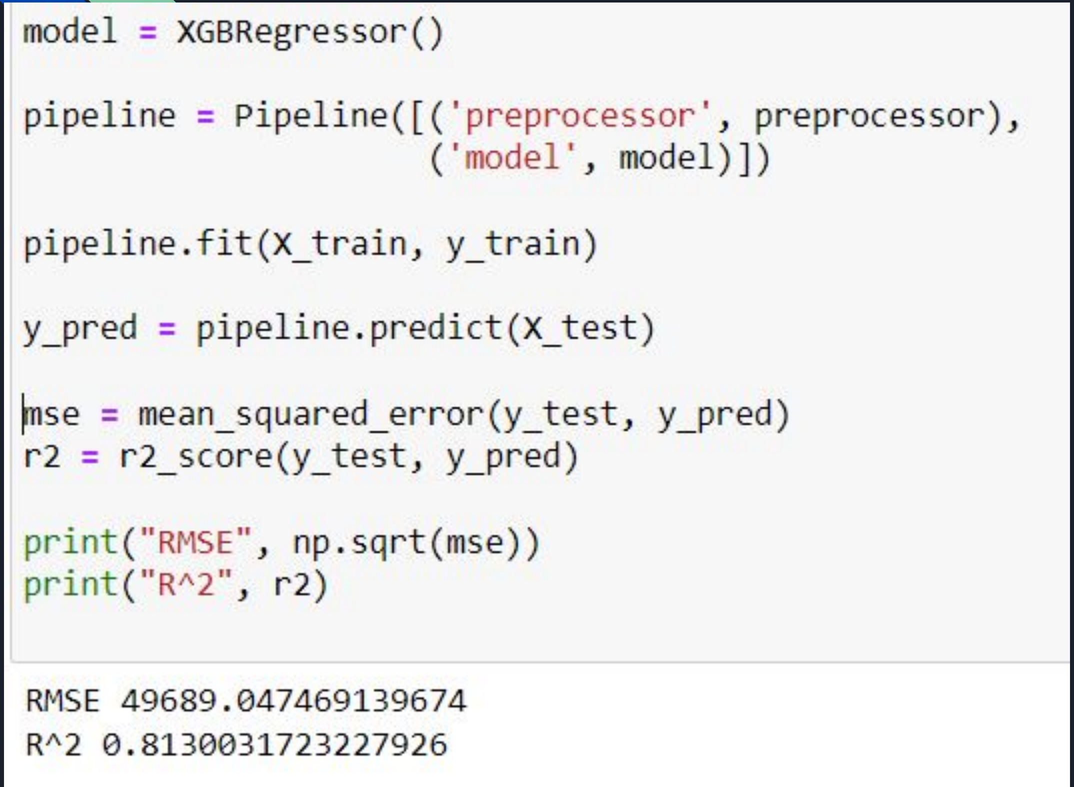
	Unnamed: 0	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
Unnamed: 0	1.00	-0.11	0.08	-0.18	0.06	0.03	0.02	0.03	0.07	0.07
longitude	-0.11	1.00	-0.92	-0.11	0.04	0.07	0.10	0.06	-0.02	-0.05
latitude	0.08	-0.92	1.00	0.01	-0.04	-0.07	-0.11	-0.07	-0.08	-0.14
housing_median_age	-0.18	-0.11	0.01	1.00	-0.36	-0.32	-0.30	-0.31	-0.11	0.11
total_rooms	0.06	0.04	-0.04	-0.36	1.00	0.93	0.86	0.92	0.20	0.13
total_bedrooms	0.03	0.07	-0.07	-0.32	0.93	1.00	0.88	0.98	-0.01	0.05
population	0.02	0.10	-0.11	-0.30	0.86	0.88	1.00	0.91	0.00	-0.03
households	0.03	0.06	-0.07	-0.31	0.92	0.98	0.91	1.00	0.01	0.06
median_income	0.07	-0.02	-0.08	-0.11	0.20	-0.01	0.00	0.01	1.00	0.69
median_house_value	0.07	-0.05	-0.14	0.11	0.13	0.05	-0.03	0.06	0.69	1.00

Preprocess :



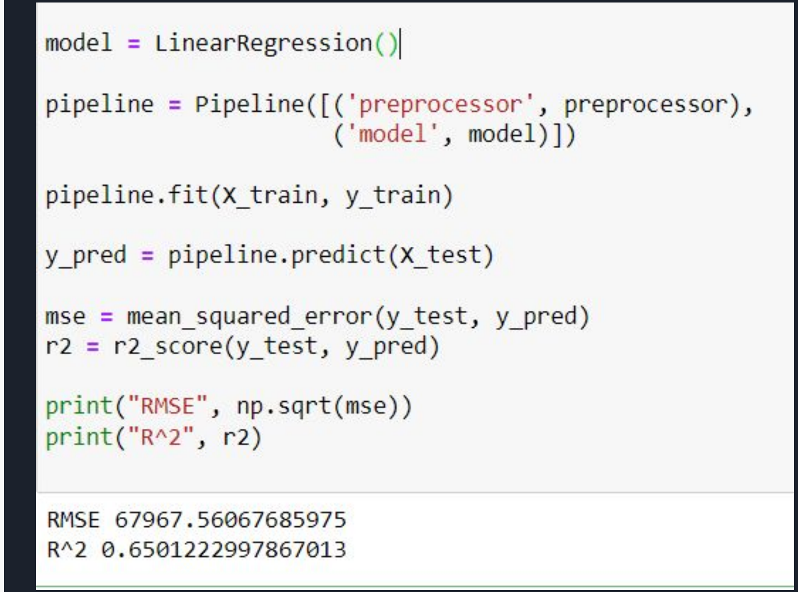
```
X = df.drop(columns=["median_house_value", 'Unnamed: 0'])
y = df['median_house_value']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

numerical_pipe = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', RobustScaler())])

categorical_pipe = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore', sparse_output=False))])
```

Le choix du modèle:



```
model = XGBRegressor()

pipeline = Pipeline([('preprocessor', preprocessor),
                    ('model', model)])

pipeline.fit(X_train, y_train)

y_pred = pipeline.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("RMSE", np.sqrt(mse))
print("R^2", r2)
```

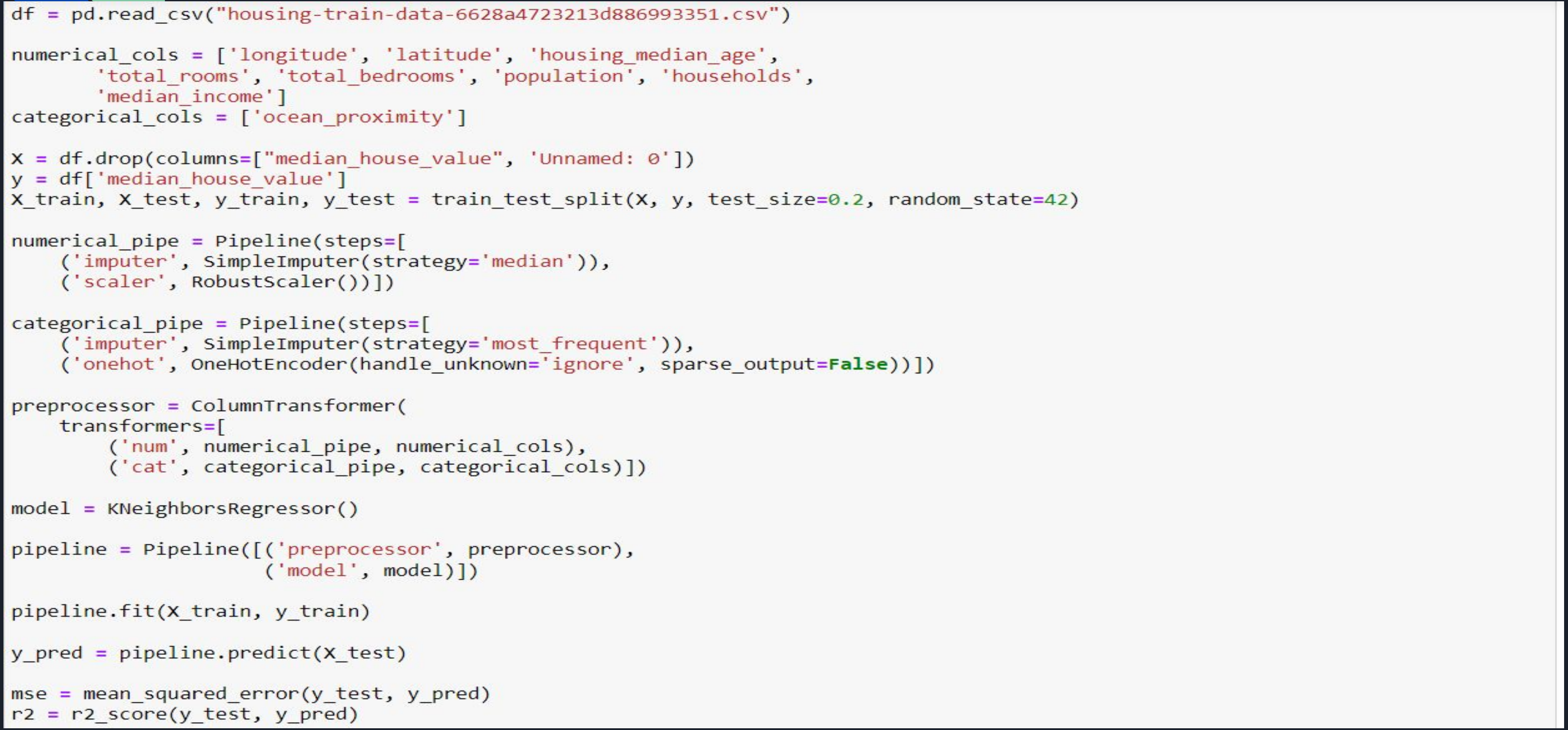
```
RMSE 49689.047469139674
R^2 0.8130031723227926
```



```
model = LinearRegression()

pipeline = Pipeline([('preprocessor', preprocessor),
                    ('model', model)])

pipeline.fit(X_train, y_train)

y_pred = pipeline.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("RMSE", np.sqrt(mse))
print("R^2", r2)
```

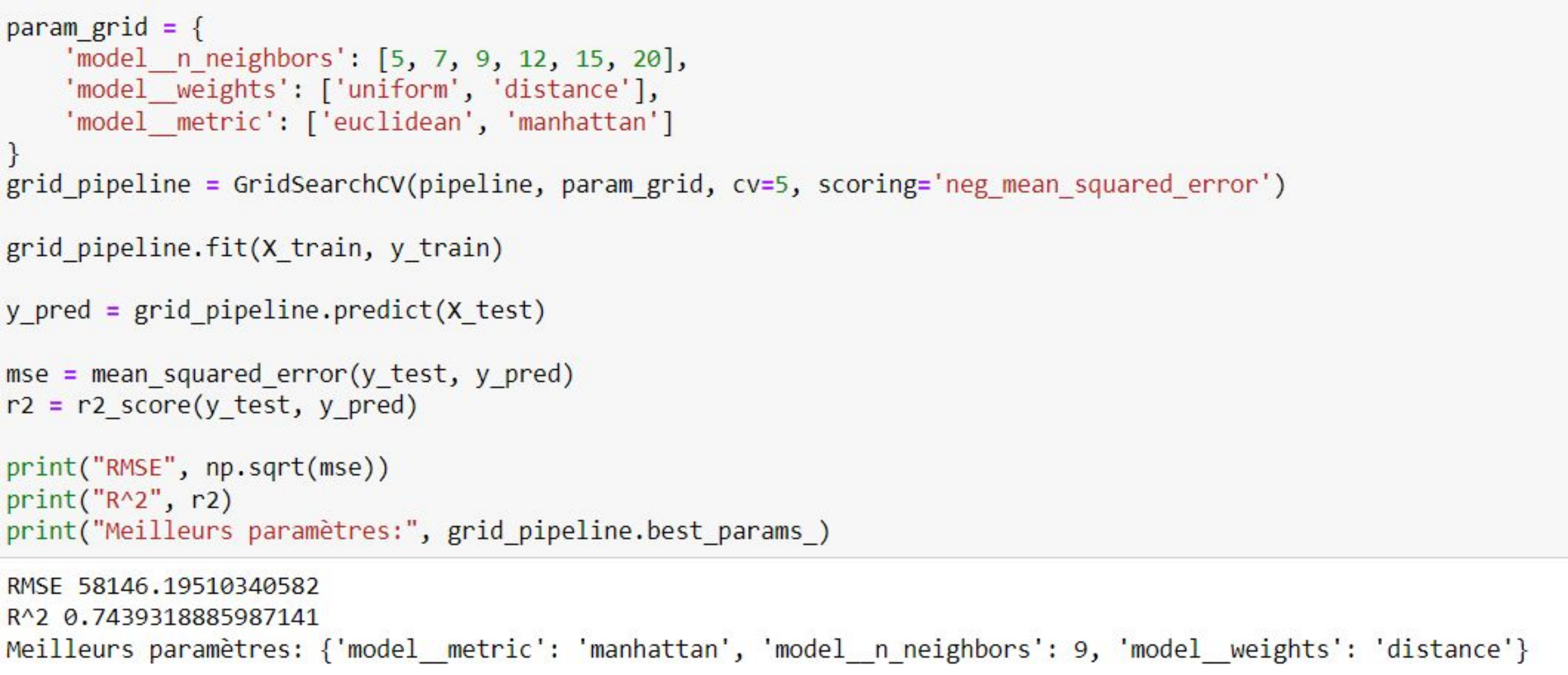
```
RMSE 67967.56067685975
R^2 0.6501222997867013
```

Model final et pipeline:



```
df = pd.read_csv("housing-train-data-6628a4723213d886993351.csv")

numerical_cols = ['longitude', 'latitude', 'housing_median_age',
       'total_rooms', 'total_bedrooms', 'population', 'households',
       'median_income']
categorical_cols = ['ocean_proximity']

X = df.drop(columns=["median_house_value", 'Unnamed: 0'])
y = df['median_house_value']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

numerical_pipe = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', RobustScaler())])

categorical_pipe = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore', sparse_output=False))])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_pipe, numerical_cols),
        ('cat', categorical_pipe, categorical_cols)])

model = KNeighborsRegressor()

pipeline = Pipeline([('preprocessor', preprocessor),
                     ('model', model)])

pipeline.fit(X_train, y_train)

y_pred = pipeline.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

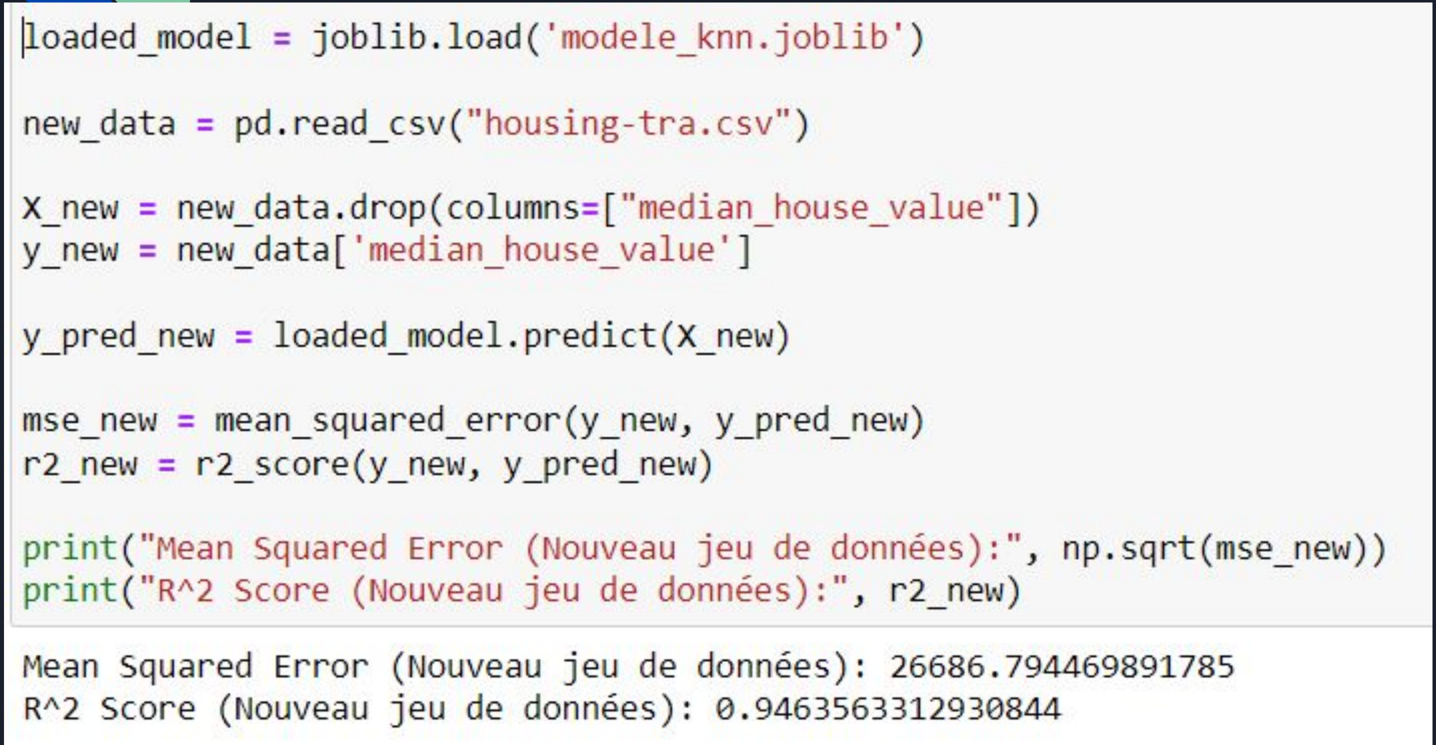
Gridsearch:



```
param_grid = {
    'model__n_neighbors': [5, 7, 9, 12, 15, 20],
    'model__weights': ['uniform', 'distance'],
    'model__metric': ['euclidean', 'manhattan']
}
grid_pipeline = GridSearchCV(pipeline, param_grid, cv=5, scoring='neg_mean_squared_error')

grid_pipeline.fit(X_train, y_train)

y_pred = grid_pipeline.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("RMSE", np.sqrt(mse))
print("R^2", r2)
print("Meilleurs paramètres:", grid_pipeline.best_params_)
```

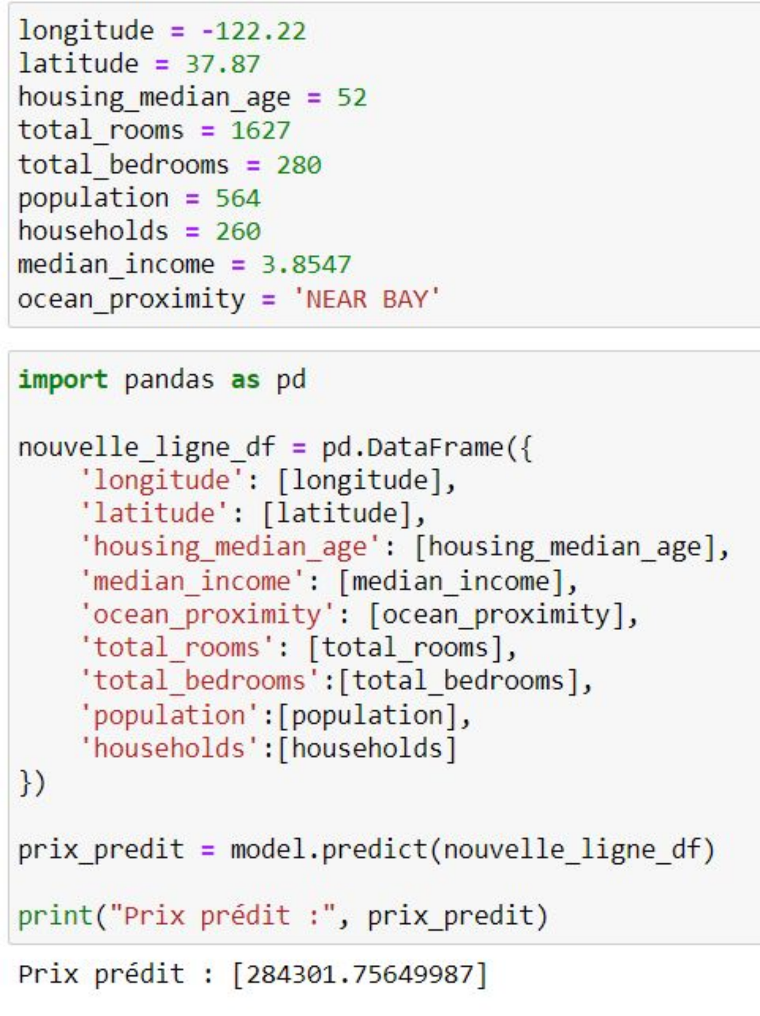
```
RMSE 58146.19510340582
R^2 0.7439318885987141
Meilleurs paramètres: {'model__metric': 'manhattan', 'model__n_neighbors': 9, 'model__weights': 'distance'}
```

Joblib et utilisation du modèle:



```python
loaded_model = joblib.load('modele_knn.joblib')

new_data = pd.read_csv("housing-tra.csv")

X_new = new_data.drop(columns=["median_house_value"])
y_new = new_data['median_house_value']

y_pred_new = loaded_model.predict(X_new)

mse_new = mean_squared_error(y_new, y_pred_new)
r2_new = r2_score(y_new, y_pred_new)

print("Mean Squared Error (Nouveau jeu de données):", np.sqrt(mse_new))
print("R^2 Score (Nouveau jeu de données):", r2_new)
```

```
Mean Squared Error (Nouveau jeu de données): 26686.794469891785
R^2 Score (Nouveau jeu de données): 0.9463563312930844
```



```
longitude = -122.22
latitude = 37.87
housing_median_age = 52
total_rooms = 1627
total_bedrooms = 280
population = 564
households = 260
median_income = 3.8547
ocean_proximity = 'NEAR BAY'
```

```
import pandas as pd

nouvelle_ligne_df = pd.DataFrame({
    'longitude': [longitude],
    'latitude': [latitude],
    'housing_median_age': [housing_median_age],
    'median_income': [median_income],
    'ocean_proximity': [ocean_proximity],
    'total_rooms': [total_rooms],
    'total_bedrooms':[total_bedrooms],
    'population':[population],
    'households':[households]
})

prix_predit = model.predict(nouvelle_ligne_df)

print("Prix prédit :", prix_predit)
```

```
Prix prédit : [284301.75649987]
```

Planning:

Mercredi : EDA + PREPROCESS

JEUDI: TEST MODEL + IT2RATION PREPROCESS

VENDREDI: TEST MODEL PIPELINES GRIDSEARCH:

LUNDI: JOBLIB STREAMLIT SLIDES GIT