Data Intake Report

Name: <Paris Housing Price Prediction>
Report date: <28/05/2021>
Internship Batch:< LISUM09>
Version:<1.0>
Data intake by:<Fahad Alothman>
Data intake reviewer:<NA>
Data storage location: <https://www.kaggle.com/datasets/mssmartypants/paris-housing-price-prediction>

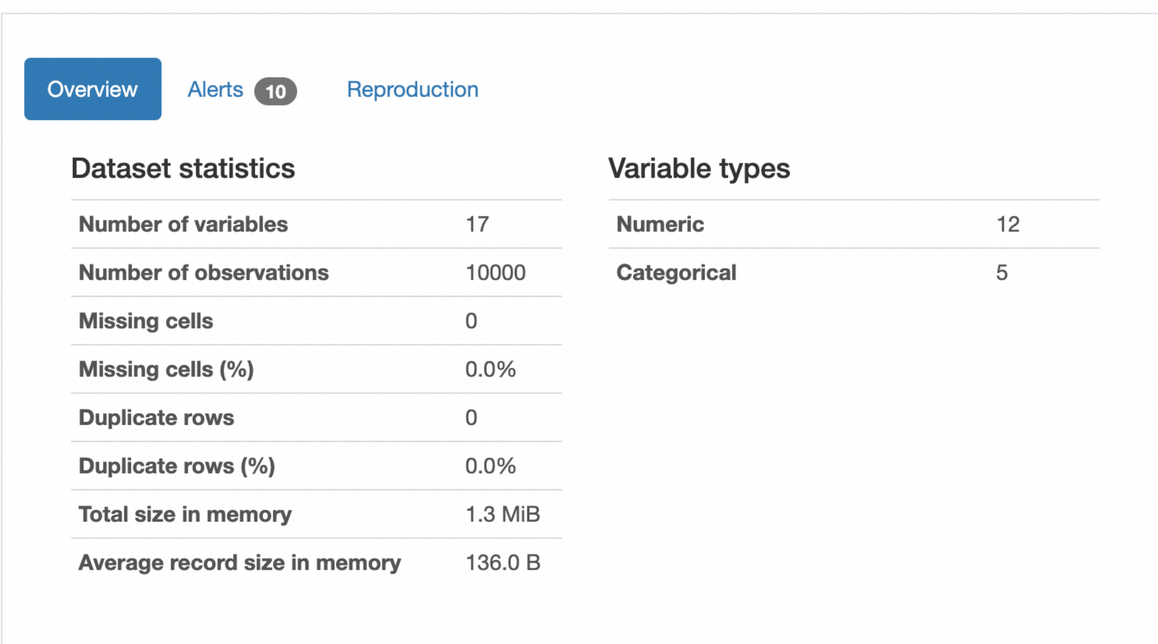
Tabular data details:

Total number of observations	10,000
Total number of files	1
Total number of features	17
Base format of the file	ParisHousing.csv
Size of the data	633 KB

Proposed Approach:

- Mention approach of dedup validation (identification):

There were no missing, inconsistent nor redundant rows in the dataset
As shown below using pandas_profiling:



Overview Alerts 10 Reproduction

Dataset statistics

Number of variables	17
Number of observations	10000
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	1.3 MiB
Average record size in memory	136.0 B

Variable types

Numeric	12
Categorical	5

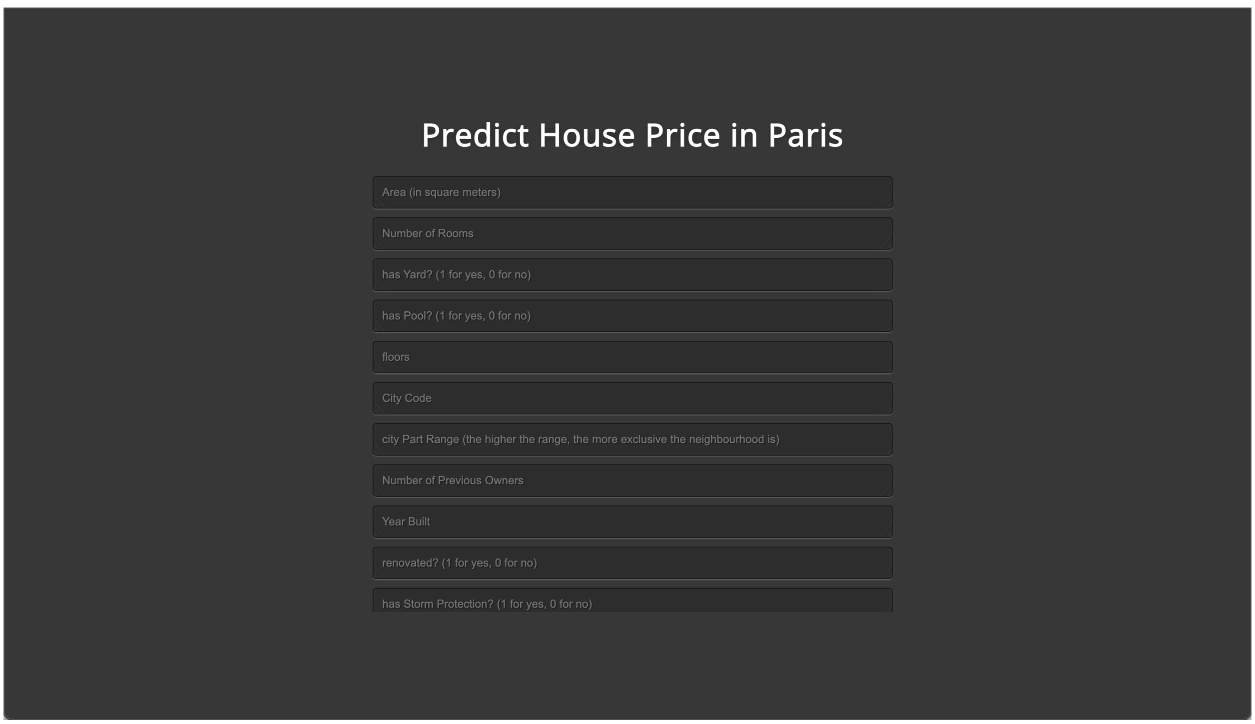
- Mention your assumptions (if you assume any other thing for data quality analysis)

1. Only 8 out of the 16 Features were required to build the linear regression Model.
2. The Root Mean Square Error chosen was 1891. That might seem high but if we take into account mean of the feature we want to predict (the price of the house) and the standard deviation, which are 4993448.0 & 2877424.0 respectively, we can safely assume that an RMSE of 1891 is appropriate and would be effective in building an accurate Model.

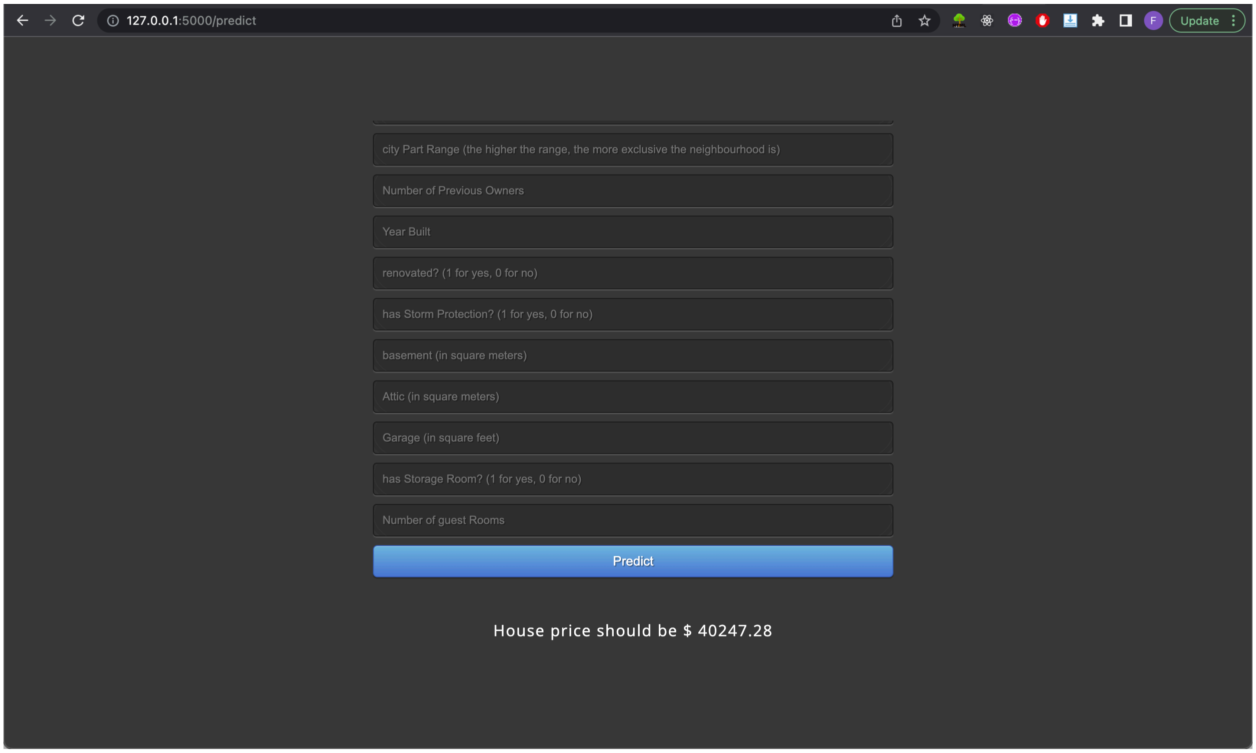
Deployment Steps:

1. Deploying the flask App

A. The user is expected to enter all 16 features of the house to predict the price (Note: I think we can reduce the inputs only to 8. But to more accurately predict the price, we decided to use all 16 features)



B. The prediction is then displayed as shown below:



2. Building and Saving the Model

This notebook tests a linear regression to determine houses' price considering different sets of features. To select features I have used Regressive Feature Elimination and tested what number of features gives the lowest RMSE.

```
#importing packages and reading the data
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pandas_profiling import ProfileReport

from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score, mean_squared_error

from sklearn import preprocessing

from sklearn.model_selection import train_test_split

from sklearn.feature_selection import RFE
```

```
db = pd.read_csv('ParisHousing.csv')
```

Since there are no missing values in any of the columns. It is safe to assume that the data requires no cleaning

Data Splitting

```
#train/test split
train_set, test_set = train_test_split(db, test_size=0.2,
random_state=42)

#isolate our target variable
label_train = train_set["price"].copy()
labels_test = test_set["price"].copy()

train_set = train_set.drop(['price'], axis = 1)
test_set = test_set.drop(['price'], axis = 1)
```

Define the linear regression:

```
linear_reg = LinearRegression()
```

We have 16 features; now let's go through all of them and see what number of features offers the best, lowest Root Mean Square Error (RMSE) on the training set:

```
#testing RFE on TRAIN
for i in range(1,17):
    rfe_i = RFE(linear_reg, n_features_to_select=i)
    rfe_i = rfe_i.fit(train_set, label_train)
    predictions_rfe_i = rfe_i.predict(train_set)
```

```
    lin_mse_rfe_i = mean_squared_error(label_train, predictions_rfe_i)
    lin_rmse_rfe_i = np.sqrt(lin_mse_rfe_i)
    print(i," ",lin_rmse_rfe_i)
```

```
1   2855176.4457531
2   2855048.4776618956
3   2855012.002018303
4   2467.1223910352737
5   2466.8849866509313
6   1897.5464224157847
7   1892.035689877902
8   1891.9109805320215
9   1891.8071559605323
10   1891.5722240673242
11   1891.5512686360325
12   1891.5332151041148
13   1891.3481676397107
14   1891.2033410323843
15   1891.0741278598962
16   1890.8705243290804
```

we can see that the change in RMSE somewhat stagnates at 8 with a value of approx. 1891

```
round(db['price'].describe())
```

```
count       10000.0
mean      4993448.0
std       2877424.0
min         10314.0
25%       2516402.0
50%       5016180.0
75%       7469092.0
max      10006771.0
Name: price, dtype: float64
```

with the mean = 4993448.0 and standard deviation = 2877424.0 having an RMSE of 1891 would be acceptable.

So, we will keep only 8 features to see how the RMSE of both the training and testing sets will turn out

```
#train set
rfe = RFE(linear_reg, n_features_to_select=8)
rfe = rfe.fit(train_set, label_train)
predictions_rfe = rfe.predict(train_set)
lin_mse_rfe = mean_squared_error(label_train, predictions_rfe)
lin_rmse_rfe = np.sqrt(lin_mse_rfe)
lin_rmse_rfe
```

```
1891.9109805320215
```

```
import pickle
```

```python
# Saving model to disk
pickle.dump(rfe, open('model.pkl', 'wb'))

# Loading model to compare the results
model = pickle.load(open('model.pkl', 'rb'))

#test set
rfe_test = model.fit(test_set, labels_test)
predictions_test_rfe = rfe_test.predict(test_set)
lin_mse_test_rfe = mean_squared_error(labels_test,
predictions_test_rfe)
lin_rmse_test_rfe = np.sqrt(lin_mse_test_rfe)
lin_rmse_test_rfe
```

```
1914.6371473885847
```

Since the RMSE of the training and testing sets are very similar (with a difference of 22.72) we can say that the model was trained well.

Now, to verify, we need to check the difference between using 8 and all 16 features and how the reduction performed was effective or not

```python
#train set
rfe = RFE(linear_reg, n_features_to_select=16)
rfe = rfe.fit(train_set, label_train)
predictions_rfe = rfe.predict(train_set)
lin_mse_rfe = mean_squared_error(label_train, predictions_rfe)
lin_rmse_rfe = np.sqrt(lin_mse_rfe)
lin_rmse_rfe
```

```
1890.8705243290804
```

```python
#test set
rfe_test = rfe.fit(test_set, labels_test)
predictions_test_rfe = rfe_test.predict(test_set)
lin_mse_test_rfe = mean_squared_error(labels_test,
predictions_test_rfe)
lin_rmse_test_rfe = np.sqrt(lin_mse_test_rfe)
lin_rmse_test_rfe
```

```
1912.5106463415534
```

Since the difference between using 8 and 16 features is at most 2.

We can safely assume that the reduction performed was effective.