House Prices Random Forest Project

Data Set

Dataset containing information of real estate listings in the US. Dataset was obtained from Kaggle at https://www.kaggle.com/datasets/ahmedshahriarsakib/usa-real-estate-dataset, original data obtained from https://www.realtor.com/.

Dataset contains columns:

- brokered_by: Encoded broker/agency
- status: Property sale status
- price: House price
- bed: Number of bedrooms
- bath: Number of bathrooms
- acre_lot: Total land size in acres
- street: Encoded street address
- city: City address
- state: State address
- zip_Code: Zip code of house
- house_size: Size of living space in square feet
- prev_sold_date: Previously sold date

prev_sold_date column was removed to make dataset smaller due to memory restrictions. For the same reason, the data was sorted by brokered_by in ascending order and only the first 300,000 rows were used for this project.

Rows with null values were also removed.

Findings

From performing operations on this dataset, I have found that the random forest model used is very innacurate, with an accuracy score of 0.0002. This model is essentially useless for predicting house prices based on the information in the dataset. This could potentially be improved if all of the dataset could be used but that was not possible in my case. Due to the format of the results, I was unable to display the confusion matrix or provide values for precision, recall, specificity or f1.

```
# import all required libraries

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# read data from file
```

```
df = pd.read_csv('housePrices.csv')
df = df.dropna()
```

```
C:\Users\Seán\AppData\Local\Temp\ipykernel_18020\1170301636.py:3:
DtypeWarning: Columns (1,7,8) have mixed types. Specify dtype option
on import or set low_memory=False.
  df = pd.read_csv('housePrices.csv')
```

```
print(df)
```

```
        brokered_by    status      price  bed  bath  acre_lot
street  \
1               0.0  for_sale   279900.0  3.0   2.0      0.20
1623254.0
2               0.0  for_sale   265000.0  4.0   3.0      0.22
296128.0
6               2.0  for_sale   510000.0  4.0   2.0      0.12
263715.0
7               2.0  for_sale   279900.0  4.0   2.0      0.19
1236322.0
8               4.0  for_sale   480000.0  4.0   4.0      0.13
885124.0
...             ...       ...        ...  ...   ...       ...
..
299979      27103.0  for_sale   369000.0  3.0   2.0      5.51
1135052.0
299989      27104.0  for_sale   439900.0  3.0   3.0      2.23
398086.0
299990      27104.0  for_sale   595000.0  4.0   2.0      2.05
729885.0
299995      27104.0  for_sale   189000.0  3.0   2.0      0.87
20535.0
299998      27104.0  for_sale  1225000.0  4.0   2.0     17.03
986675.0

                    city         state  zip_code  house_size
1              Savannah       Georgia   31419.0      1728.0
2              Savannah       Georgia   31419.0      1487.0
6            Minneapolis     Minnesota   55414.0      4058.0
7       Columbia Heights     Minnesota   55421.0      2556.0
8          Wesley Chapel       Florida   33543.0      2484.0
...                  ...           ...       ...         ...
299979         Lake Como  Pennsylvania   18437.0      2800.0
299989           Shohola  Pennsylvania   18458.0      4411.0
299990    Dingmans Ferry  Pennsylvania   18328.0      1840.0
299995          Tamiment  Pennsylvania   18371.0      1497.0
299998           Milford  Pennsylvania   18337.0      2896.0

[143018 rows x 11 columns]
```

```
# select target column and feature columns
# encode categorical data

x = df.drop(columns=['price'])
y = df['price']
x = pd.get_dummies(x, drop_first=True)
y = pd.get_dummies(y, drop_first=True)

# randomly select data to split into training and test data, 30% test
data

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =
0.3, random_state = 26)

model = RandomForestClassifier(n_estimators=10, max_depth=10,
random_state=26)
model.fit(x_train, y_train)
```

```
RandomForestClassifier(max_depth=10, n_estimators=10, random_state=26)
```

```
y_pred = model.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)

print(accuracy)
```

```
0.00027968116347364006
```