Predicción de precio en propiedades de Airbnb

Mario Emilio Jiménez Vizcaíno
A01173359@itesm.mx
Ingeniería en Tecnologías Computacionales
Tecnológico de Monterrey

Franco Daniel Pérez Reyes
A00822080@itesm.mx
Ingeniería en Mecatrónica
Tecnológico de Monterrey

Alberto Dávila Almaguer
A01196987@itesm.mx
Ingeniería en Innovación y Desarrollo
Tecnológico de Monterrey

Carlos Andrés Luna Leyva
A00826148@itesm.mx
Ingeniería en Tecnologías Computacionales
Tecnológico de Monterrey

ABSTRACT

En los últimos años han surgido plataformas en las cuales anfitriones pueden publicitar y contratar el arriendo de sus propiedades con sus huéspedes; anfitriones y huéspedes pueden valorarse mutuamente, como referencia para futuros usuarios. Una de las primeras, y la más exitosa en Estados Unidos es AirBnB, fundada en el año 2008. En este proyecto se utilizarán técnicas de aprendizaje automático con el fin de predecir y optimizar una serie de métricas sobre propiedades, utilizando un dataset de más de 70 mil propiedades listadas desde noviembre de 2008 hasta octubre 2017 en esta plataforma.

1 INTRODUCCIÓN

El precio de cualquier producto o servicio se ve afectado por variables como oferta, demanda, inflación en el país, etc. Para bienes inmuebles esto es mucho más complejo, ya que entran en consideración factores como la zona donde se encuentra la propiedad, el espacio total que abarca, el número de habitaciones, etc. Cada una de estas variables puede o no afectar a las demás y juntas caracterizan un sistema digno de ser analizado.

2 CONCEPTOS PREVIOS

- Conocimientos básicos de estadística
- Programación básica en el lenguaje Python
- Conocimientos sobre librerías como `scikit-learn`, `pandas` y `numpy`

3 METODOLOGÍA

TODO

4 RESULTADOS

TODO

5 CONCLUSIONES

TODO

REFERENCES

A CÓDIGO DE EJECUCIÓN DE LOS REGRESORES

```python
import pathlib
import numpy as np
import pandas as pd
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor, VotingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor

df_base = pd.read_csv('airbnb_clean.csv').set_index("id")
df_ammenities = pd.read_csv('airbnb_amenities_clean.csv').set_index("id")
df = pd.concat([df_base, df_ammenities], axis=1)

X = df.drop(columns=["log_price", "host_since", "host_has_profile_pic", "city",
                     "host_identity_verified", "property_type", "room_type"])
Y = df["log_price"]

X["cleaning_fee"] = X["cleaning_fee"].astype(int)
X["instant_bookable"] = X["instant_bookable"].astype(int)
X[df_ammenities.columns] = X[df_ammenities.columns].astype(int)

rf_mses = []
gb_mses = []
lr_mses = []
kn_mses = []
vr_mses = []

for i in range(30):
    print(f"Calculating scores for iteration {i+1}")
    X_train, X_test, y_train, y_test = train_test_split(
        X, Y, test_size=0.2, random_state=i)

    random_forest = RandomForestRegressor(n_estimators=100, n_jobs=-1)
    random_forest.fit(X_train, y_train)
    rf_pred = random_forest.predict(X_test)
    rf_mses.append(mean_squared_error(y_test, rf_pred))

    gradient_boosting = GradientBoostingRegressor(n_estimators=150)
    gradient_boosting.fit(X_train, y_train)
    gb_pred = gradient_boosting.predict(X_test)
    gb_mses.append(mean_squared_error(y_test, gb_pred))

    linear_reg = LinearRegression(fit_intercept=True)
    linear_reg.fit(X_train, y_train)
    lr_pred = linear_reg.predict(X_test)
    lr_mses.append(mean_squared_error(y_test, lr_pred))

    k_neighbors = KNeighborsRegressor(n_neighbors=30, n_jobs=-1)
    k_neighbors.fit(X_train, y_train)
    kn_pred = k_neighbors.predict(X_test)
    kn_mses.append(mean_squared_error(y_test, kn_pred))

    voting_reg = VotingRegressor([("RF", random_forest), ("GB", gradient_boosting),
                                  ("KN", k_neighbors), ("LN", linear_reg)], n_jobs=-1)
    voting_reg.fit(X_train, y_train)
    vr_pred = voting_reg.predict(X_test)
    vr_mses.append(mean_squared_error(y_test, vr_pred))

    print(f"{rf_mses[i]:2.5f} {gb_mses[i]:2.5f}", end=" ")
    print(f"{lr_mses[i]:2.5f} {kn_mses[i]:2.5f} {vr_mses[i]:2.5f}")
```

B SALIDA DE EJECUCIÓN DE LOS REGRESORES

```
Calculating scores for iteration 1
0.17583 0.18862 0.29061 0.27962 0.19436
Calculating scores for iteration 2
0.17881 0.18835 0.28611 0.27449 0.19290
Calculating scores for iteration 3
0.17301 0.18280 0.27537 0.26925 0.18600
Calculating scores for iteration 4
0.17623 0.18881 0.28657 0.28166 0.19390
Calculating scores for iteration 5
0.17483 0.18737 0.28715 0.27676 0.19200
Calculating scores for iteration 6
0.17366 0.18453 0.28149 0.27784 0.19029
Calculating scores for iteration 7
0.17639 0.18611 0.28709 0.27331 0.19206
Calculating scores for iteration 8
0.17819 0.18926 0.29059 0.28183 0.19590
Calculating scores for iteration 9
0.17487 0.18631 0.28020 0.27405 0.18994
Calculating scores for iteration 10
0.17268 0.18663 0.28243 0.27379 0.18992
Calculating scores for iteration 11
0.17527 0.18710 0.28620 0.27405 0.19206
Calculating scores for iteration 12
0.17455 0.18893 0.28500 0.27107 0.19063
Calculating scores for iteration 13
0.17531 0.18802 0.28308 0.27220 0.19208
Calculating scores for iteration 14
0.17485 0.18806 0.28952 0.27267 0.19275
Calculating scores for iteration 15
0.17623 0.18632 0.28253 0.27538 0.19105
Calculating scores for iteration 16
0.17335 0.18552 0.27961 0.27192 0.18910
Calculating scores for iteration 17
0.17291 0.18649 0.28193 0.27358 0.18973
Calculating scores for iteration 18
0.17437 0.18559 0.28341 0.27621 0.19148
Calculating scores for iteration 19
0.17343 0.18511 0.28152 0.27081 0.18863
Calculating scores for iteration 20
0.17776 0.18946 0.28510 0.27851 0.19392
Calculating scores for iteration 21
0.18095 0.19456 0.29053 0.27785 0.19696
Calculating scores for iteration 22
0.17118 0.18228 0.27875 0.26795 0.18694
Calculating scores for iteration 23
0.17113 0.18675 0.28304 0.27240 0.19027
Calculating scores for iteration 24
0.17124 0.18415 0.28199 0.27108 0.18764
Calculating scores for iteration 25
0.17631 0.18797 0.28490 0.27225 0.19187
Calculating scores for iteration 26
0.17590 0.18882 0.28670 0.27830 0.19293
Calculating scores for iteration 27
0.17297 0.18495 0.27996 0.26918 0.18770
Calculating scores for iteration 28
0.17261 0.18690 0.28336 0.27616 0.19117
Calculating scores for iteration 29
0.17671 0.19054 0.28790 0.27424 0.19418
Calculating scores for iteration 30
0.17552 0.18500 0.28215 0.27992 0.19099
```

C CÓDIGO DE PRUEBA DE WILCOXON

```python
from pathlib import Path
import numpy as np
import pandas as pd
from scipy.stats import ranksums

mses_dict = {}

for mses in Path('mses').iterdir():
    if mses.is_file():
        mses_dict[mses.stem] = np.loadtxt(mses)

if len(mses_dict) == 0:
    print("No MSEs files found")
    print("Run `generate_mses.py` file first")

matrix_greater = []
matrix_less = []

for mses1 in mses_dict.values():
    row_greater = []
    row_less = []

    for mses2 in mses_dict.values():
        row_greater.append(
            ranksums(mses1, mses2, alternative="greater").pvalue)
        row_less.append(ranksums(mses1, mses2, alternative="less").pvalue)

    matrix_greater.append(row_greater)
    matrix_less.append(row_less)

df_greater = pd.DataFrame(
    matrix_greater, index=mses_dict.keys(), columns=mses_dict.keys())
df_less = pd.DataFrame(
    matrix_less, index=mses_dict.keys(), columns=mses_dict.keys())

pd.options.display.float_format = "{:.14f}".format
print("Wilcoxon rank sums test with a greater hypothesis")
print(df_greater)
print()

print("Wilcoxon rank sums test with a less hypothesis")
print(df_less)
```

D SALIDA DE EJECUCIÓN DE PRUEBA DE WILCOXON

D.1 Wilcoxon rank sums test with a greater hypothesis

	RandomForest	GradientBoosting	LinearRegression	KNeighbors	VotingRegressor
RandomForest	0.5	0.99999999998564	0.99999999998564	0.99999999998564	0.99999999998564
GradientBoosting	0.00000000001436	0.5	0.99999999998564	0.99999999998564	0.99999991000660
LinearRegression	0.00000000001436	0.00000000001436	0.5	0.00000000022015	0.00000000001436
KNeighbors	0.00000000001436	0.00000000001436	0.99999999977985	0.5	0.00000000001436
VotingRegressor	0.00000000001436	0.00000008999340	0.99999999998564	0.99999999998564	0.5

D.2 Wilcoxon rank sums test with a less hypothesis

	RandomForest	GradientBoosting	LinearRegression	KNeighbors	VotingRegressor
RandomForest	0.5	0.00000000001436	0.00000000001436	0.00000000001436	0.00000000001436
GradientBoosting	0.99999999998564	0.5	0.00000000001436	0.00000000001436	0.00000008999340
LinearRegression	0.99999999998564	0.99999999998564	0.5	0.99999999977985	0.99999999998564
KNeighbors	0.99999999998564	0.99999999998564	0.00000000022015	0.5	0.99999999998564
VotingRegressor	0.99999999998564	0.99999991000660	0.00000000001436	0.00000000001436	0.5