Predicting Airbnb Prices Using Ridge Regression

Santiago Primera Escobedo

ITCS-3156-002

https://github.com/santip5/Regression-on-Airbnb-prices

Introduction

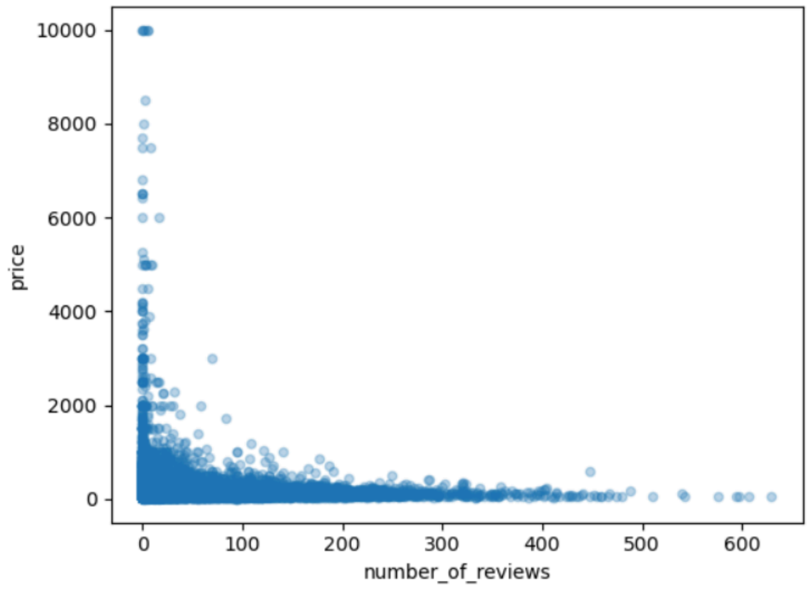
This project will analyze Airbnb listings in NYC and build regression models to predict nightly rental prices. Understanding what drives prices is important for both hosts and customers to make sure that they are providing and getting the best value. Using a 2019 NYC Airbnb dataset, I will train two regression models to evaluate how well characteristics such as room type, location, and availability predict price.

Data

The dataset (kaggle) contains approximately 47000 listings with information on room type, neighborhood, price, host name, availability, among others.

```
[15]: airbnb_df.plot.scatter(x="number_of_reviews", y="price", alpha=0.3)
```

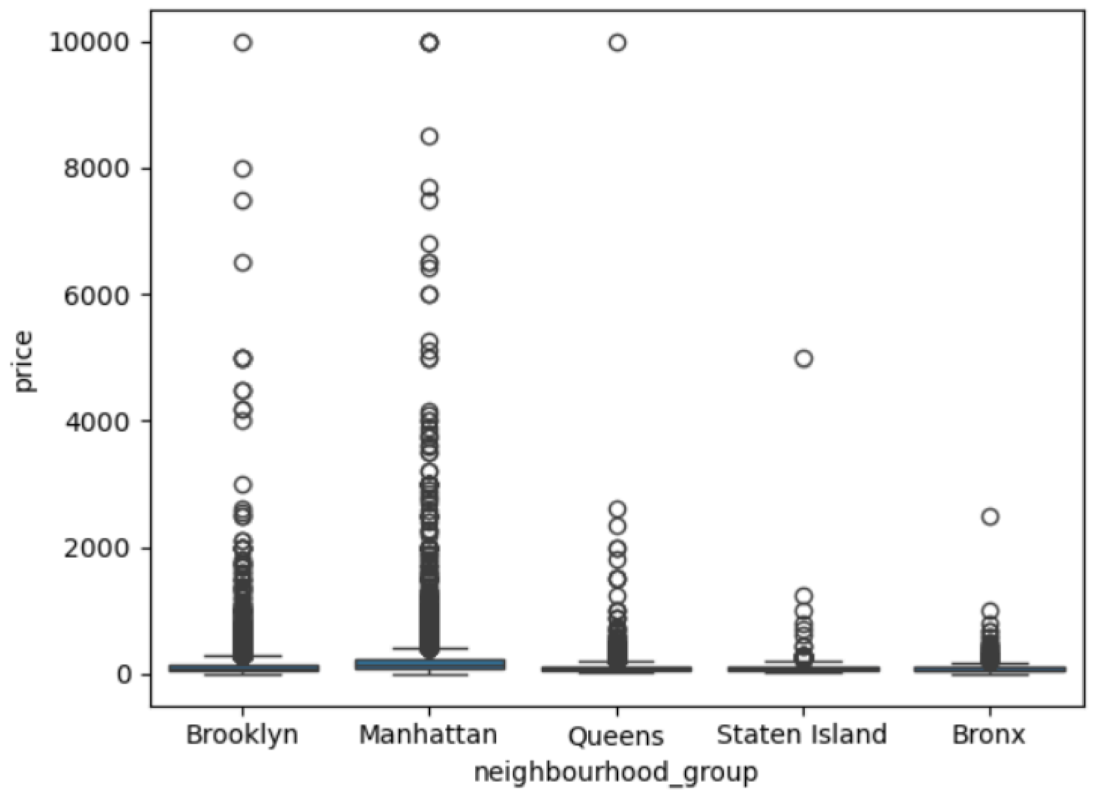
```
[15]: <Axes: xlabel='number_of_reviews', ylabel='price'>
```



There isn't a clear linear relationship between price and # of reviews, but cheaper listings tend to accumulate more reviews.

```
[25]: sns.boxplot(data=airbnb_df, x="neighbourhood_group", y="price")
```

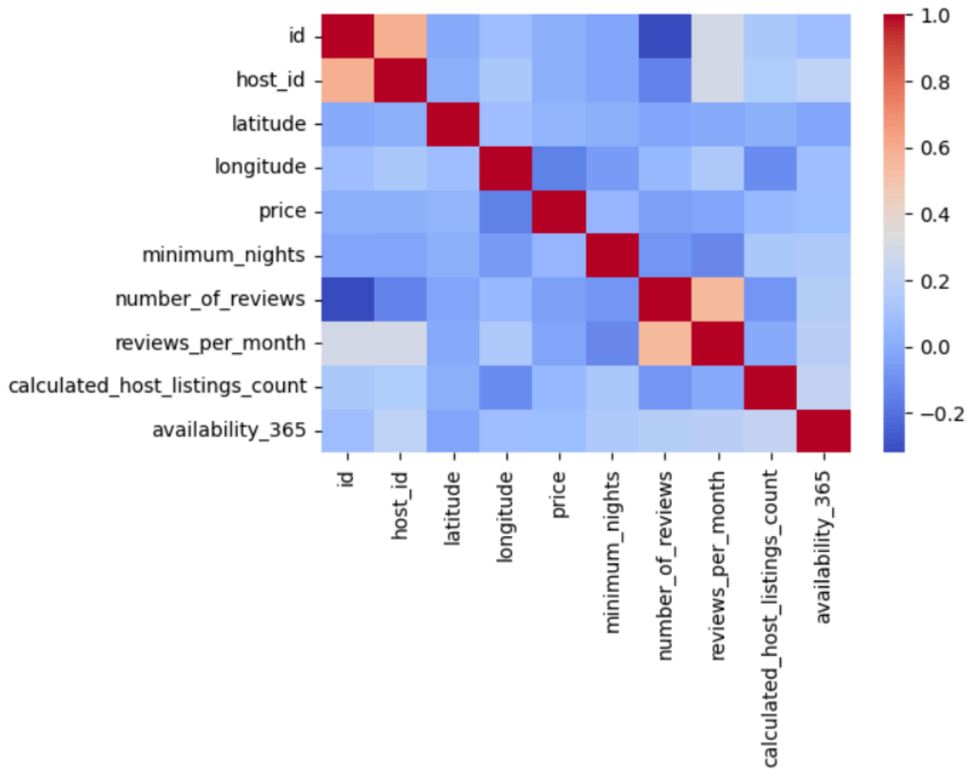
```
[25]: <Axes: xlabel='neighbourhood_group', ylabel='price'>
```



Manhattan is the most expensive borough, followed by Brooklyn.

```
[23]: plt.figure(figsize=(6,4))
      sns.heatmap(airbnb_df.corr(numeric_only=True), annot=False, cmap="coolwarm")
```

```
[23]: <Axes: >
```



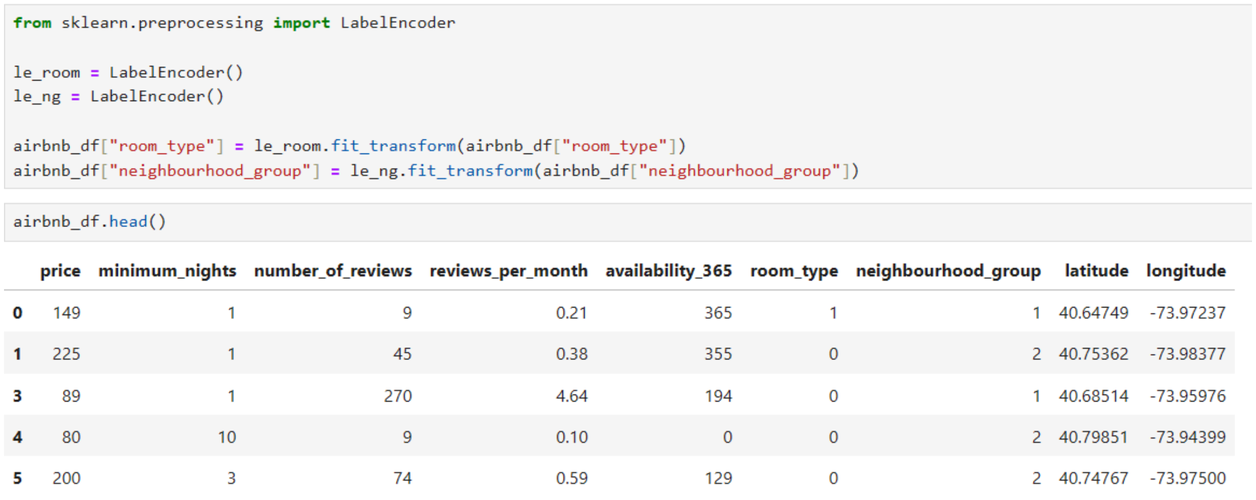
Correlation between numeric predictors is low overall, suggesting weak linear relationship between features.

```
airbnb_df = airbnb_df[[
    "price",
    "minimum_nights",
    "number_of_reviews",
    "reviews_per_month",
    "availability_365",
    "room_type",
    "neighbourhood_group",
    "latitude",
    "longitude"
]]

airbnb_df = airbnb_df.dropna()

airbnb_df = airbnb_df[airbnb_df["price"] <= 1000]
```

```
airbnb_df.head()
```

	price	minimum_nights	number_of_reviews	reviews_per_month	availability_365	room_type	neighbourhood_group	latitude	longitude
0	149	1	9	0.21	365	Private room	Brooklyn	40.64749	-73.97237
1	225	1	45	0.38	355	Entire home/apt	Manhattan	40.75362	-73.98377
3	89	1	270	4.64	194	Entire home/apt	Brooklyn	40.68514	-73.95976
4	80	10	9	0.10	0	Entire home/apt	Manhattan	40.79851	-73.94399
5	200	3	74	0.59	129	Entire home/apt	Manhattan	40.74767	-73.97500

Dropping non-numerical features (such as host name) and null values. I'm also dropping any rows with price over 1000 since those are usually outliers/luxury rentals and may not represent the larger population.

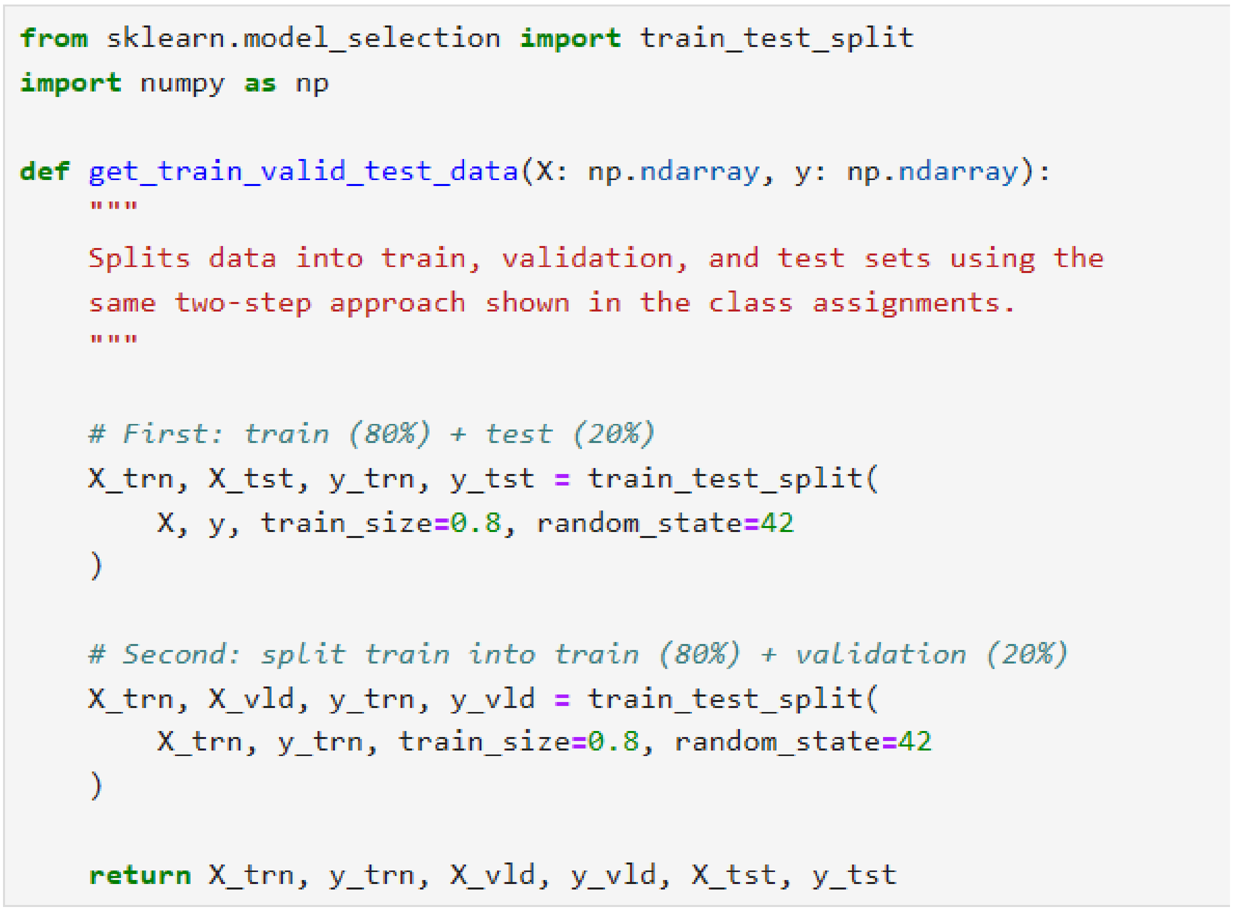


```python
from sklearn.preprocessing import LabelEncoder

le_room = LabelEncoder()
le_ng = LabelEncoder()

airbnb_df["room_type"] = le_room.fit_transform(airbnb_df["room_type"])
airbnb_df["neighbourhood_group"] = le_ng.fit_transform(airbnb_df["neighbourhood_group"])
```

```python
airbnb_df.head()
```

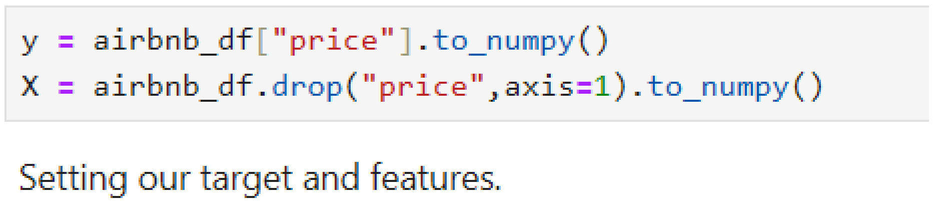
	price	minimum_nights	number_of_reviews	reviews_per_month	availability_365	room_type	neighbourhood_group	latitude	longitude
0	149	1	9	0.21	365	1	1	40.64749	-73.97237
1	225	1	45	0.38	355	0	2	40.75362	-73.98377
3	89	1	270	4.64	194	0	1	40.68514	-73.95976
4	80	10	9	0.10	0	0	2	40.79851	-73.94399
5	200	3	74	0.59	129	0	2	40.74767	-73.97500

Enconding remaining non-numerical features that are relevant to price.



```python
from sklearn.model_selection import train_test_split
import numpy as np

def get_train_valid_test_data(X: np.ndarray, y: np.ndarray):
    """
    Splits data into train, validation, and test sets using the
    same two-step approach shown in the class assignments.
    """

    # First: train (80%) + test (20%)
    X_trn, X_tst, y_trn, y_tst = train_test_split(
        X, y, train_size=0.8, random_state=42
    )

    # Second: split train into train (80%) + validation (20%)
    X_trn, X_vld, y_trn, y_vld = train_test_split(
        X_trn, y_trn, train_size=0.8, random_state=42
    )

    return X_trn, y_trn, X_vld, y_vld, X_tst, y_tst
```

Splitting the data into training, validation, and test sets.



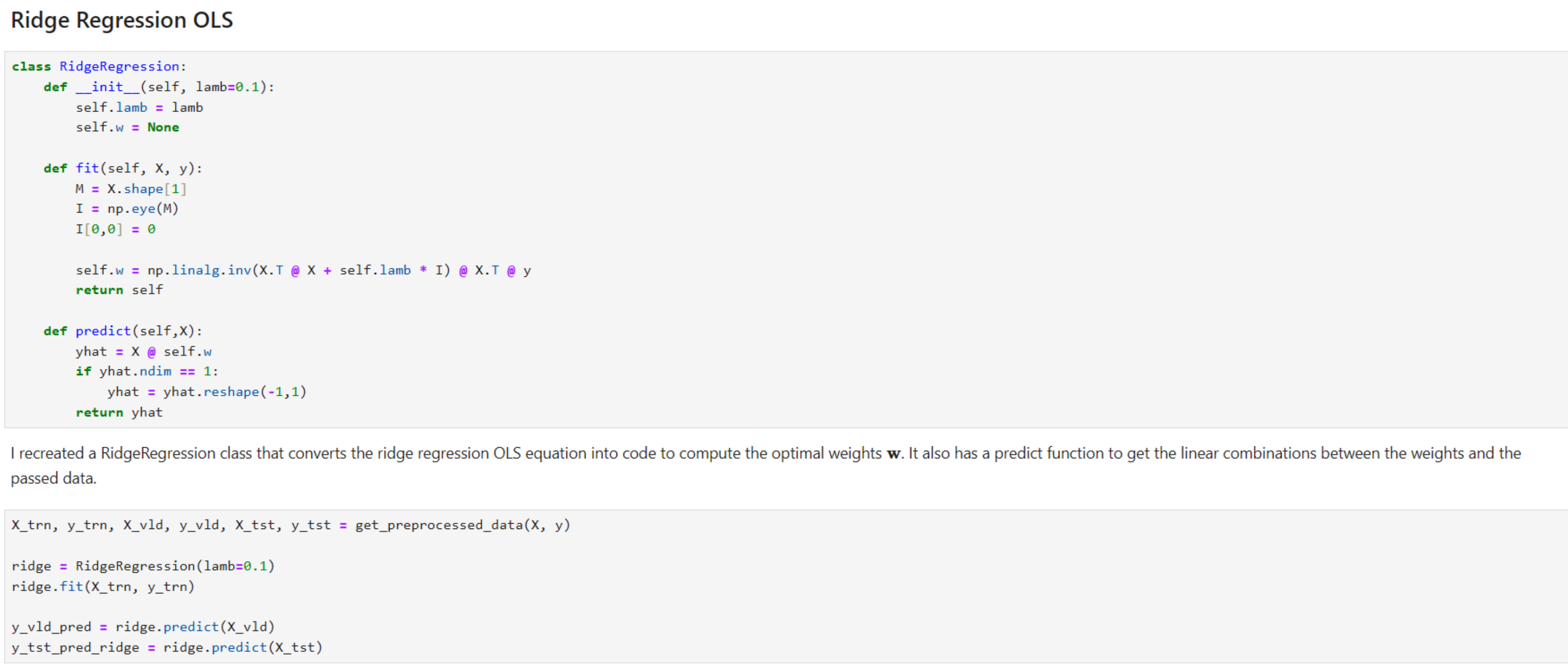
```
y = airbnb_df["price"].to_numpy()
X = airbnb_df.drop("price",axis=1).to_numpy()
```

Setting our target and features.

Methods and Results

In this project I will use two regression models, ridge regression and random forest regression. Ridge regression is good for this dataset because of its L2 regularization. Random forest regression was chosen in order to capture any relationships that aren't showcased by the linear regression model, and can provide a point of comparison.

1. **Linear Regression** - minimizes the closed form equation with a regularization term (lambda) to control overfitting.



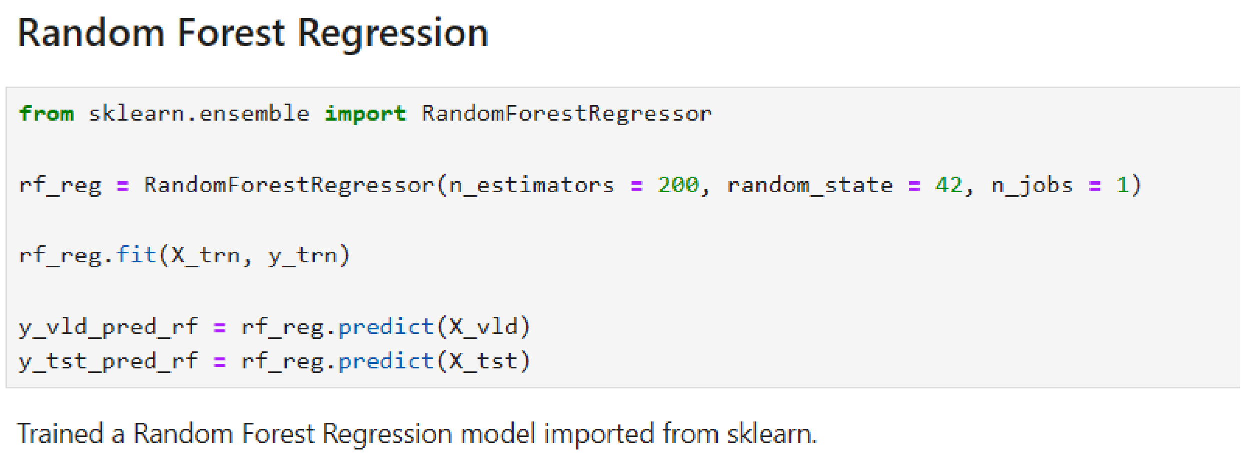
Ridge Regression OLS

```
class RidgeRegression:
    def __init__(self, lamb=0.1):
        self.lamb = lamb
        self.w = None

    def fit(self, X, y):
        M = X.shape[1]
        I = np.eye(M)
        I[0,0] = 0

        self.w = np.linalg.inv(X.T @ X + self.lamb * I) @ X.T @ y
        return self

    def predict(self,X):
        yhat = X @ self.w
        if yhat.ndim == 1:
            yhat = yhat.reshape(-1,1)
        return yhat
```

I recreated a RidgeRegression class that converts the ridge regression OLS equation into code to compute the optimal weights **w**. It also has a predict function to get the linear combinations between the weights and the passed data.

```
X_trn, y_trn, X_vld, y_vld, X_tst, y_tst = get_preprocessed_data(X, y)

ridge = RidgeRegression(lamb=0.1)
ridge.fit(X_trn, y_trn)

y_vld_pred = ridge.predict(X_vld)
y_tst_pred_ridge = ridge.predict(X_tst)
```

2. **Random Forest Regression -** uses many decision trees to capture the different relationships between the features and price.

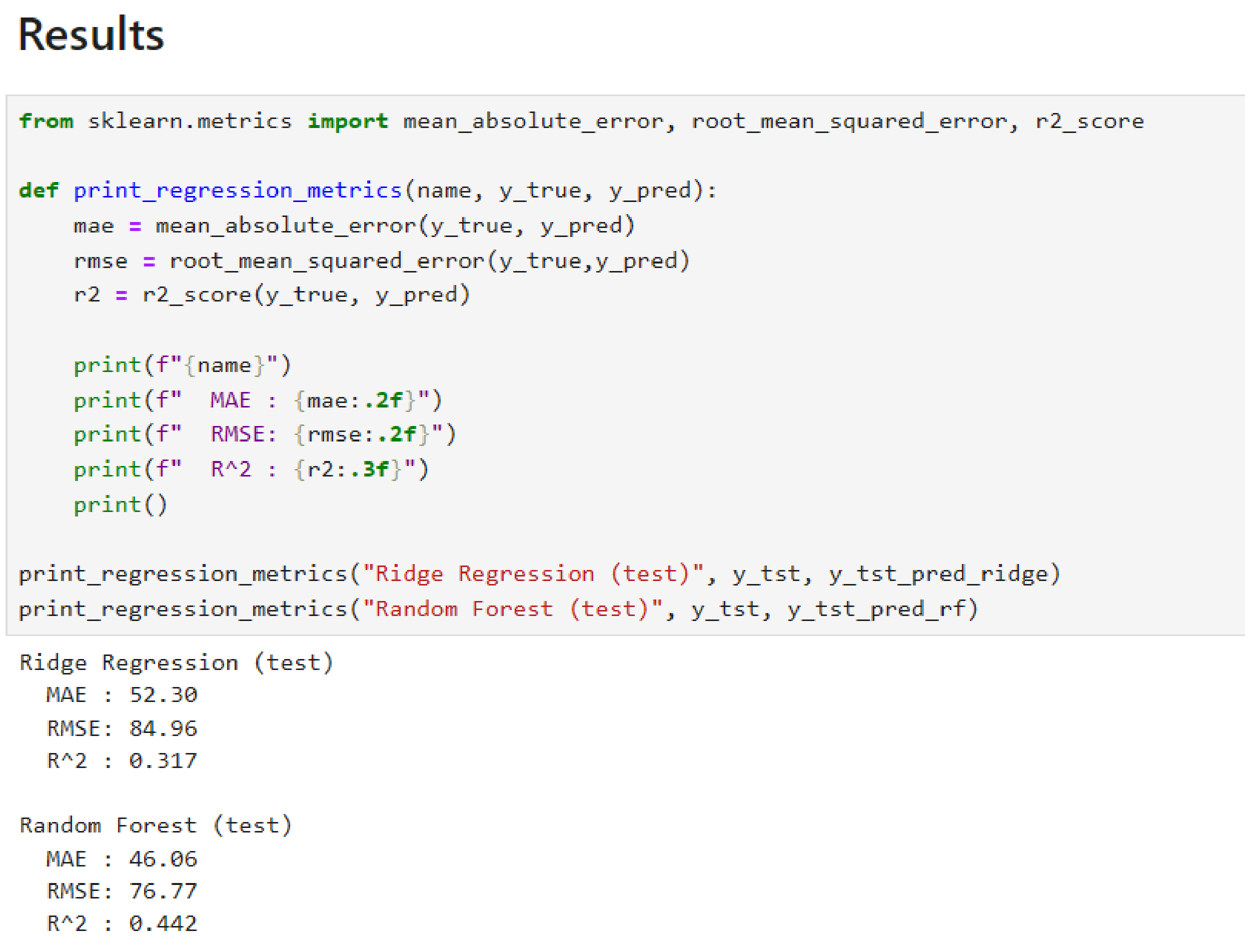


Random Forest Regression

```python
from sklearn.ensemble import RandomForestRegressor

rf_reg = RandomForestRegressor(n_estimators = 200, random_state = 42, n_jobs = 1)

rf_reg.fit(X_trn, y_trn)

y_vld_pred_rf = rf_reg.predict(X_vld)
y_tst_pred_rf = rf_reg.predict(X_tst)
```

Trained a Random Forest Regression model imported from sklearn.

Results



Results

```python
from sklearn.metrics import mean_absolute_error, root_mean_squared_error, r2_score

def print_regression_metrics(name, y_true, y_pred):
    mae = mean_absolute_error(y_true, y_pred)
    rmse = root_mean_squared_error(y_true,y_pred)
    r2 = r2_score(y_true, y_pred)

    print(f"{name}")
    print(f"  MAE : {mae:.2f}")
    print(f"  RMSE: {rmse:.2f}")
    print(f"  R^2 : {r2:.3f}")
    print()

print_regression_metrics("Ridge Regression (test)", y_tst, y_tst_pred_ridge)
print_regression_metrics("Random Forest (test)", y_tst, y_tst_pred_rf)
```

```
Ridge Regression (test)
  MAE : 52.30
  RMSE: 84.96
  R^2 : 0.317

Random Forest (test)
  MAE : 46.06
  RMSE: 76.77
  R^2 : 0.442
```

The Ridge Regression model was implemented from scratch using the closed-form solution with L2 regularization, while the Random Forest model was trained using the scikit-learn library. Both models predicted Airbnb listing price, and their performance was compared using MAE, RMSE, and R^2.

From the results, Random Forest performs better than Ridge Regression on all evaluation metrics. It has a lower MAE (46.06 vs 52.30) and lower RMSE (76.77 vs 84.96), meaning its predictions are closer to the true prices on average., The R^2 score is also higher (.442 vs .317) indicating that it explains more of the variance in listing prices.

Despite the differences, the common thread is still that both models struggled to predict the price accurately. This is likely due to more complex relationships between the features or other features such as amenities, photos available, cleaning fee, etc. not being available for the dataset. The low correlation between our features that was observed in the earlier heatmap further corroborates these results.

Conclusion

In this project I explored predicting Airbnb prices using two regression models, a Ridge regression model implemented from scratch and a Random Forest model imported from scikit-learn. Through the results I learned that the relationships between the variables and price was not enough to explain the price variation in this dataset. Random Forest performed better likely due to being more complex, however both models still fell well below having a strong predictive accuracy. If this dataset is to be remade, additional features could be included in order to better capture the whole ecosystem.

Overall this project helped refine my understanding of the ML pipeline by allowing me to experiment with a dataset I wasn’t familiar with and starting from scratch. It also highlighted some challenges I’ll have to overcome such as becoming more familiar with which models would work better for specific situations and data not always telling the full story.

Works Cited

Dgomonov. (2019, August 12). New York City airbnb open data. Kaggle. https://www.kaggle.com/datasets/dgomonov/new-york-city-airbnb-open-data?resource=download

Randomforestregressor. scikit. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

AI Acknowledgement

No AI was used in the making of this project, online resources such as the scikit-learn website and other library resources were used.