Pattern Recognition
Project(1): Apartment Rent Prediction
Milestone 1 Report
Team_ID: CS_29

لجين عماد محمد صلاح	2021170413
مارينا نبيل كامل بنيامين	2021170432
جنى محمود مرسى محمود	2021170143
لطيفه احمد حنفى عبدالفتاح	2021170414
مريم مصطفى عبد اللاه	2021170514

• Preprocessing Techniques

1. Missing Data Handling:

- Null values in categorical columns such as **'pets_allowed'** and **'amenities'** were replaced with **'none'** to signify the absence of these features.
- For numeric columns like **'bathrooms'** and **'bedrooms'** missing values were filled with the **mode**, as these are discrete categories. Additionally, these columns were converted from float to integer for data integrity.
- For continuous data like **'latitude'** and **'longitude'** missing values were replaced with the **mean** as they are continuous data.
- The **'cityname'** and **'state'** columns their missing values were updated first from the new columns which are **'city_name_from_body'** and **'state_from_body' that** we extract from **'body'** column using regular expression on it.

Then remaining missing on them were replaced with **mode** , The most common city or state was used to fill any remaining missing values.

- The column **'address'** we apply **geocoding preprocessing** to fill nulls in it by deriving the address from **'longitude'** and **'latitude'.**

2. Feature Engineering and Encoding

- **Label Encoding** was applied to transform categorical columns (**'price_type', 'currency', 'category', 'fee'**) into numeric labels.It assigns a unique integer value to each category within a column.Label encoding is appropriate when there are relatively few unique categories in a column such as in (**'price_type', 'currency', 'category', 'fee',**).
- **One-hot encoding** was used for columns like **'has_photo'**, **'pets_allowed', 'amenities ' and 'source'.** One-hot encoding converts a categorical variable into one or more

columns, where each column represents a unique category . and after encoding, the original columns were dropped to avoid redundancy. We use this as there are more than 2 unique variables but less than 50. In two columns **'pets_allowed'** and **' amenities '** we separated by (,) in each row. We create a column for each unique variable. In **'source'** due to having multiple unique variables but except the top two the rest had very low frequency. We grouped them into single category called ‘Other’ before performing One-Hot encoding.

- **CatBoost encoding** was applied for the **'state', 'cityname', 'body', 'title'**, and **'address'** columns. This approach transformed these text-based columns into numeric format. CatBoost encoding is a form of target encoding which is a technique used in data preprocessing to transform categorical features based on the **target variable (price_display)**. Unlike label

encoding and one-hot encoding, target encoding uses the statistical relationship between a categorical feature and the target variable to create a numeric representation. Which is helps handle categorical variables with high cardinality (a large number of unique categories). The difference is Catboost is normalized compared to target encoding which catboost is used here for higher performance.

3. **Data Cleaning and Transformation**:
    - The **'price_display'** column, initially a string, was stripped of non-numeric characters and we convert to integers. We remove from it the comma(,) and dollar sign($). We drop '**price**' column and we put '**price_display**' after cleaning as the target column '**Y'.**
    - For column **'body'** we extract information from it and create two new columns which are **'city_name_from_body'** and **'state_from_body'** using **Regular**

expression applied on **'body'** column. We use these new columns we extract to replace nulls in **'cityname'** column and **'state'** column.

- For column **'title'** we extract information from it and create two new columns which are **'bedrooms_from_title'** and **'addresses_from_title'** using **Regular expression** applied on **'title'** column.

4. Apply NLP

- We find that there is columns such that data[‘body’] which contain text we decide to preprocess it by applying NLP to collect the most words that affect on the target data[‘price’].We do first

  1-preprocesses the text data in the 'body' column of our Data by tokenizing each row.

  2-converting tokens to lowercase,

3-removing stopwords

4- and joining the filtered tokens back into a string.

5-We uses TF-IDF vectorization to transform the preprocessed text data into a TF-IDF matrix.

After Applying TF-IDF on the preprocessed column we try to calculate the sum of TF-IDF scores for each word across all documents in corpus, then sorts these word scores in descending order to identify the most important words. And to select the top 50 words with the highest cumulative TF-IDF scores and print them.

This is the result of the 60 top words

```
Top 60 most important words:
available     606.381675
range         598.732225
rates         589.531261
rental        585.500461
units         560.959301
unit          558.481889
rent          556.214067
located       550.230348
beds          537.984679
apartment     386.359138
one           360.424784
two           315.542305
features      307.767899
include       305.333959
txmonthly     275.364824
fitness       233.325611
balcony       227.635758
deck          227.314766
three         216.986891
street        193.212196
bedrooms      191.343304
bd            188.731278
studio        184.552145
air           123.698301
living        122.592280
controlled    119.023724
surface       118.962291
laundry       116.221391
income        112.773281
avenue        112.648291
garbage       111.956380
san           110.718008
feet          109.466361
facilities    105.093109
parking       101.639744
new            98.018812
home           97.335778
water          95.864636
pool           93.123475
park           90.328699
dishwasher     87.339754
road           87.037075
city           86.491900
area           86.156659
wamonthly      86.055809
access         85.406318
pet            83.967847
ilmonthly      83.193128
heat           82.150342
dtype: float64
```

Then we try to find the correlation between

`Two, community, features, include`

and the target **(data['Price'])**

This is the final results

```
two          -0.078374
community    -0.012413
features     -0.105371
include      -0.095018
dtype: float64
```

From the results we don’t found that there is not a new features that has correlation more than 0.3 which we use to feature selection step

So we ignore them.

5.Standardization (Feature Scaling):

Standard scaling is used to standardize the range of the data by centering the data around the mean and scaling it to unit variance. This helps ensure that all features contribute equally to the model's learning process.

•Analysis of dataset:

This is the number of columns in the data and its info :

```
Data columns (total 22 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   id              9000 non-null   int64  
 1   category        9000 non-null   object 
 2   title           9000 non-null   object 
 3   body            9000 non-null   object 
 4   amenities       9000 non-null   object 
 5   bathrooms       9000 non-null   int64  
 6   bedrooms        9000 non-null   int64  
 7   currency        9000 non-null   object 
 8   fee             9000 non-null   object 
 9   has_photo       9000 non-null   object 
 10  pets_allowed    9000 non-null   object 
 11  price           9000 non-null   int64  
 12  price_display   9000 non-null   object 
 13  price_type      9000 non-null   object 
 14  square_feet     9000 non-null   int64  
 15  address         9000 non-null   object 
 16  cityname        9000 non-null   object 
 17  state           9000 non-null   object 
 18  latitude        9000 non-null   float64
 19  longitude       9000 non-null   float64
 20  source          9000 non-null   object 
 21  time            9000 non-null   int64  
dtypes: float64(2), int64(6), object(14)
```

	id	bathrooms	bedrooms	price	square_feet	latitude	longitude	time
count	9.000000e+03	9000.000000	9000.000000	9000.000000	9000.000000	9000.000000	9000.000000	9.000000e+03
mean	5.623668e+09	1.346000	1.346000	1487.286222	947.138667	37.676890	-94.778612	1.574906e+09
std	7.007402e+07	0.581081	0.581081	1088.561190	668.806214	5.513124	15.763098	3.755142e+06
min	5.508654e+09	1.000000	1.000000	200.000000	106.000000	21.315500	-158.022100	1.568744e+09
25%	5.509250e+09	1.000000	1.000000	950.000000	650.000000	33.666975	-101.543850	1.568781e+09
50%	5.668610e+09	1.000000	1.000000	1275.000000	802.000000	38.744000	-93.737500	1.577358e+09
75%	5.668626e+09	2.000000	2.000000	1695.000000	1100.000000	41.349800	-82.474025	1.577359e+09
max	5.668663e+09	8.000000	8.000000	52500.000000	40000.000000	61.594000	-70.191600	1.577362e+09

- **Feature selection:**

  We calculate the correlation matrix between the features (columns) and a target variable '**price_display'**. Then, we select the features ('**bathrooms**','**'bedrooms'**,**'cityname'**,**'state'**, **square_feet**' and **'address'**)that have an absolute correlation coefficient greater than 0.3 with the **price_display** variable.

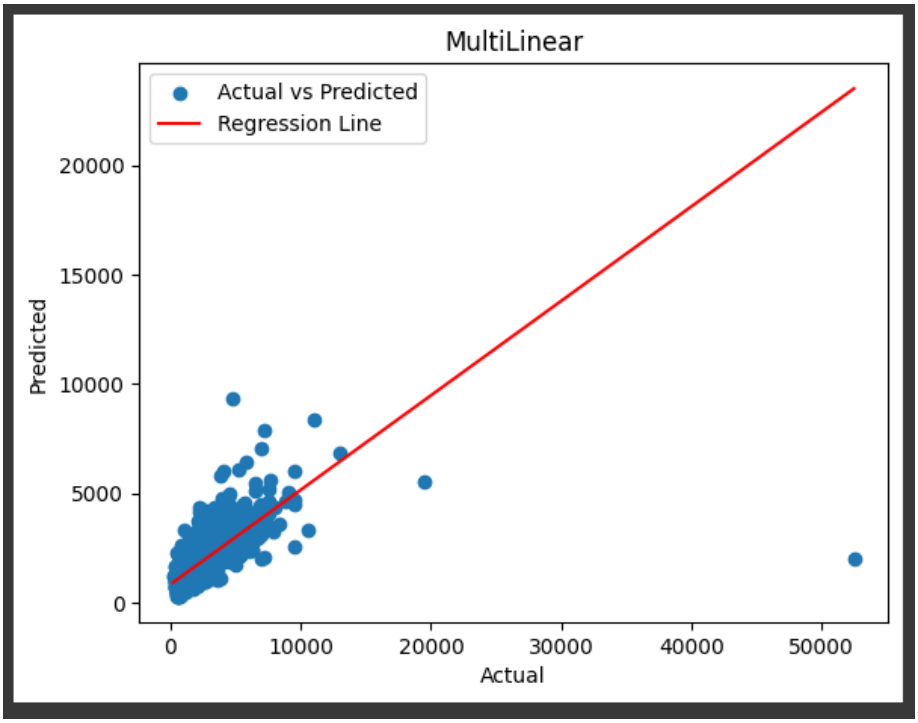
- **Size of Train and test:**

  We split the dataset into training and testing sets for both features (X) and the target variable (Y), where testing set is 20% of the total dataset.

- **Regression Techniques:**

  1-Multilevel Regression: we perform multilevel linear regression using scikit-learn's Linear Regression model on training set and testing set

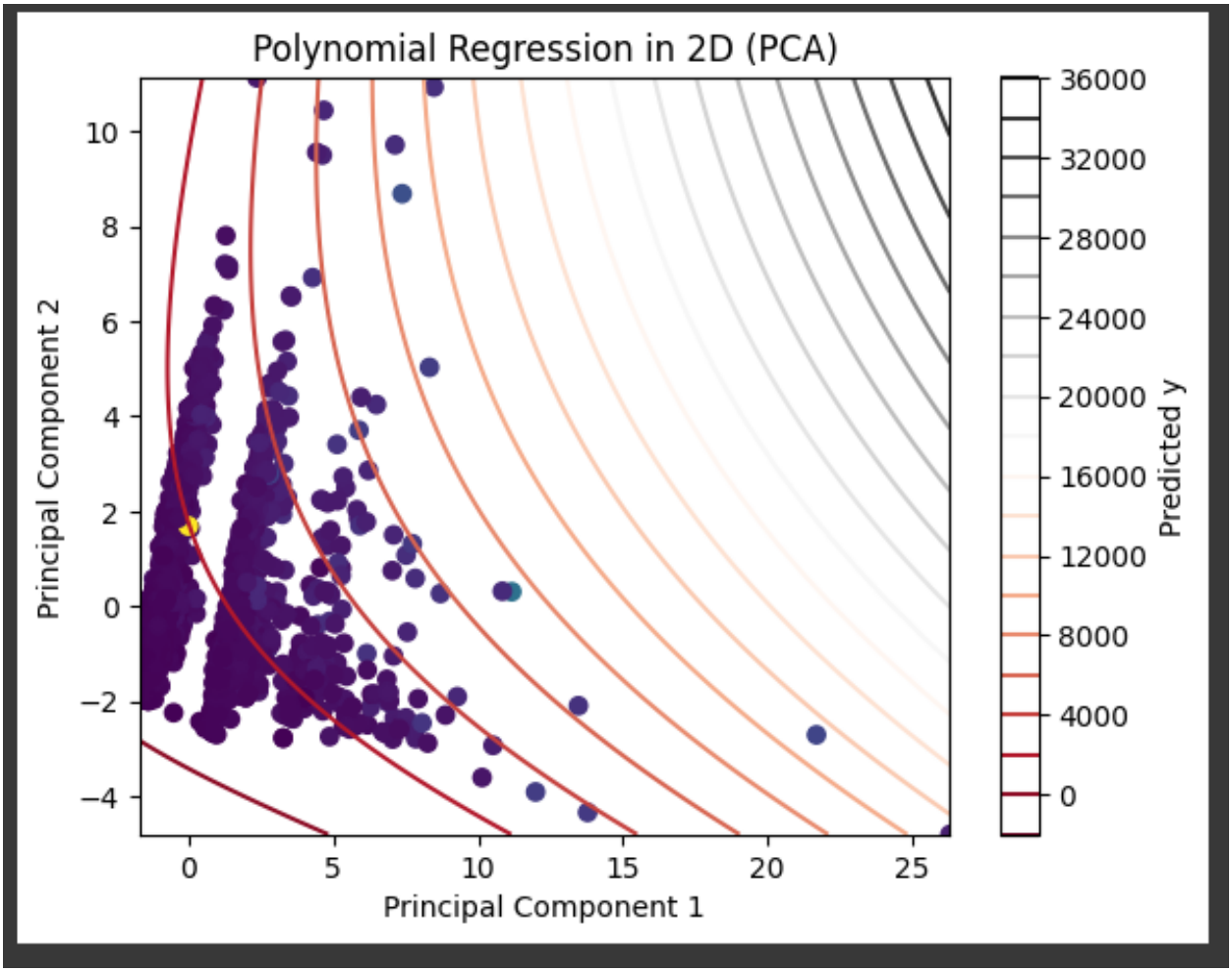
- training set: Mean Square Error = 654038.140265437 , Accuracy= 44.101605198677106
- testing set: Mean Square Error = 417465.95349291153 , Accuracy= 66.41356801962614



2-Polynomial Regression: we perform polynomial regression; we use Polynomial Features to transform the original features

(**X_train**) into polynomial features up to the specified degree (degree 2) and we perform linear regression model (**poly_model**) from scikit-learn's **linear_model** module. This model will be used to fit the polynomial features to the target variable (**y_train**) to generate predictions for the target variable.

- training set: Mean Square Error = 571602.0901095077 , Accuracy= 51.14713143604248
- testing set: Mean Square Error = 316318.7078759635 , Accuracy= 74.55117794084688



3- Random Forest Regression:

It builds multiple decision trees during training and outputs the average prediction of the individual trees. This helps to reduce overfitting and improve the accuracy of the model, we use this model with specific hyperparameters, evaluates its performance such as (n_estimators=300, max_depth=10,

min_samples_split=30, n_jobs=-1, random_state=0, oob_score=True).

- Out-of-Bag Score: 0.4980237174318204
- Mean Squared Error: 441889.0949326922
- Model Accuracy: 64.44865046922243

4-Decision Tree Regression:

builds a tree-like model that predicts the target variable by partitioning the feature space into regions and assigning a prediction value to each region based on the training data, we use this model with specific hyperparameters such as

1. **min_samples_split**: The minimum number of samples required to split an internal node (30).
2. **max_depth**: The maximum depth of the decision tree (10).
   - Mean Squared Error: 426849.7573937188
   - Model Accuracy: 65.65861186381359

5- Gradient Boosting Regression:

We use Gradient Boosting Regression to handle complex relationships between features and target variables. we use this model with specific hyperparameters such as

- **n_estimators**: The number of boosting stages or trees to be built (300)
- **min_samples_split**: The minimum number of samples required to split an internal node(20).

Mean Squared Error: 365979.16850505193

Model Accuracy: 70.55583971247748

• Used Features to create Regression Models:

After applying correlation we find that the top features to use are: [‘bathrooms’, ‘bedrooms’,

‘square_feet’, ‘address’, ‘cityname’, ‘state’] as shown in the selected columns after encoding below:

	bathrooms	bedrooms	square_feet	address	cityname	state
5250	1	1	516	1205.942260	1235.703618	1175.769961
7173	3	3	1850	1487.286222	1624.215548	1132.766296
2661	1	1	556	1487.286222	1186.021086	1299.423722
5018	1	1	675	1487.286222	1487.495249	1266.574433
3969	1	1	410	1487.286222	1306.177713	1294.699298
...	...	...	...	...	...	...
4829	2	2	1036	1487.286222	1310.095407	1531.997605
7291	1	1	900	1487.286222	1317.665194	1577.722690
1344	1	1	902	1487.286222	1018.031802	1792.752725
7293	1	1	250	1487.286222	2476.457244	1173.554192
1289	1	1	1050	1019.091873	1019.091873	1294.699298

•Improve the Results

1. In **polynomial regression** we perform hyperparameter tuning by using random search to select the best degree for polynomial regression .
2. In **Random forest regression** we use these hyperparameters to improve accuracy **(n_estimators=300, max_depth=10, min_samples_split=30, n_jobs=-1, random_state=0, oob_score=True).**

3. **In Decision Tree Regression** we use these hyperparameters to improve accuracy **min_samples_split , max_depth.**
4. **In Gradient Boosting Regression** we use these hyperparameters to improve accuracy **n_estimators , min_samples_split.**

- **Conclusion:**

1. We predict that Random forest Regression will be the best model in accuracy, and we found that **Polynomial Regression** model is

the best model in accuracy & performance and is our selected model.

- MSE For **Random Forest**: 441889.0949326922
- Model Accuracy: 64.44865046922243

- MSE For **Polynomial**: 316955.1118780659
- Model Accuracy: 74.49997726316333

2. our intuition about the importance of certain features, specifically the **number of bedrooms**, **bathrooms** and **square_feet** was validated. These features played a significant role in predicting apartment rents. We also found that there are features play a significant role in predicting and have high correlation as: **‘cityname’ , ‘address’ , ‘state’** in contrast to our first assumption of them not having high enough correlation.

In conclusion of this phase we learn that shouldn’t make assumption before analysis of

the data well and clearly and that preprocessing is the most important phase of any machine learning project.