**Capstone project on Airbnb – Rental property price prediction**

* **Table of contents**
* Introduction
* Project Structure
* Data Description
* Setup Instructions
* Data Preprocessing
* Model Training and Evaluation
* Results
* Feature Importance
* Future Work
* **Introduction**

This project aims to predict rental property prices using a range of machine learning models. The dataset used includes various features such as room type, property type, host information, and reviews, sourced from rental property listings. The main goal is to build an accurate prediction model and analyse feature importance to understand which factors influence rental prices the most.

* **Project Structure**
* data: Contains the datasets used in the project.
* notebooks: Jupyter notebooks for exploratory data analysis and model development.
* src: Source code for data preprocessing, model training, and evaluation.
* README.md: This file.
* **Data Description**

The dataset consists of the following tables

* listings.csv: Contains information about the properties.
* hosts.csv: Contains details about the property hosts.
* reviews.csv: Contains reviews of the properties.
* calendar.csv: Contains pricing and availability data for the properties.
* **Setup Instruction**

Prerequisites

* + Python
  + Pip (python installer package) install required packages.
* **Data Preprocessing**
* **Data imputation / data merging** – clean / impute / Combined `listings`, `hosts`, `reviews`, and `calendar` tables based on `listing\_id`.
* **Feature engineering** - Created new features like `log\_price`, `scaled\_price`, `host\_duration`, and regional clustering.

df\_calendar['log\_price'] = np.log(df\_calendar['price'] + 1)

* **Handling Missing Values** - Imputed missing values using mean for continuous features and mode for categorical features.

df\_concatenated.fillna(df\_concatenated.mean(), inplace=True)

* **Encoding Categorical Variables** - One-hot encoded variables like `room\_type` and `property\_type`

room\_type\_encoded = pd.get\_dummies(df\_listings['room\_type'], prefix='room\_type')

* **Normalization** - Standardized numerical features for consistent scaling

scaler = StandardScaler()

df\_calendar['scaled\_price'] = scaler.fit\_transform(df\_calendar[['price']])

* **Model Training & Evaluation**

The following machine learning models were trained and evaluated using cross-validation and performance metrics like R-squared, Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE):

* + Decision Tree
  + Gradient Boosting
  + Random Forest
  + Extra Trees
  + XGBoost
  + LightGBM

**Code for model training**

from sklearn.model\_selection import cross\_val\_score

from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

from sklearn.tree import DecisionTreeRegressor

from xgboost import XGBRegressor

from lightgbm import LGBMRegressor

**Define models**

models = {

'Decision Tree': DecisionTreeRegressor(),

'Gradient Boosting': GradientBoostingRegressor(),

'Random Forest': RandomForestRegressor(),

'Extra Trees': ExtraTreesRegressor(),

'XGBoost': XGBRegressor(),

'LightGBM': LGBMRegressor()

}

# Evaluate each model

results = {}

for model\_name, model in models.items():

cv\_results = cross\_val\_score(model, X, y, cv=5, scoring='r2')

results[model\_name] = cv\_results.mean()

# Select the best model

best\_model\_name = max(results, key=results.get)

best\_model = models[best\_model\_name]

print(f"Best Model: {best\_model\_name} with R2 Score: {results[best\_model\_name]:.4f}")

* **Results**

With individual regression of model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MODEL | MSE | R-squared | MAE | RMSE |
| Linear Regression | 7.43E+24 | -2.60E+25 | 2.66E+12 | 2.73E+12 |
| Regression Tree | 3.07E-05 | 0.9999 | 8.73E-05 | 0.0055 |
| Random Forest Regressor | 3.29E-05 | 0.9999 | 1.11E-04 | 0.0057 |
| Gradient Boosting | 0.0045 | 0.9844 | 0.0434 | 0.0668 |
| XGBoot Regressor | 9.71E-05 | 0.9997 | 0.0044 | 0.0099 |
| LightGBM Regressor | 0.0003 | 0.999 | 0.0104 | 0.0166 |

With cross validation

* Decision Tree: R2 Score = 0.9327
* Gradient Boosting: R2 Score = 0.9642
* Random Forest: R2 Score = 0.9478
* Extra Trees: R2 Score = 0.9290
* XGBoost: R2 Score = 0.9715
* LightGBM: R2 Score = 0.9730
* **Feature Importance**

The top features influencing the model predictions were determined

importances = best\_model.feature\_importances\_

feature\_names = X\_train.columns

feature\_importances = pd.DataFrame({'feature': feature\_names, 'importance': importances})

feature\_importances = feature\_importances.sort\_values(by='importance', ascending=False)

top\_5\_features = feature\_importances.head(5)

print(top\_5\_features)

* **Future Work**
* Model Optimization: Further tuning of hyperparameters to improve model performance.
* Feature Engineering: Exploration of additional features such as neighbourhood data and property amenities.
* Deployment: Develop a web application for real-time rental price predictions.