Singapore Resale Flat Prices Prediction

Project Report summarizing the data analysis, model development, and deployment process.

Problem Statement:

There are many factors that affect the selling price of resale apartments in Singapore. Hence, the objective of implementing decision tree regression is to examine the relationship between the selling price of a resale flat and its various attributes, including its distance to the Central Business District (CBD), distance to the nearest MRT station, flat size, floor level, and remaining lease duration.

Dataset:

Link: https://beta.data.gov.sg/collections/189/view

OneMap API: https://www.onemap.gov.sg/apidocs/#onemap-rest-apis

Data Preprocessing:

- The Resale Flat Prices dataset has five distinct CSV files, each representing a specific time. These time periods are 1990 to 1999, 2000 to 2012, 2012 to 2014, 2015 to 2016, and 2017 onwards. Therefore, **it is essential to merge the five distinct CSV files into a unified dataset**.

```
df = pd.concat([pd.read_csv(f) for f in glob.glob("./data/*.csv")], ignore_index=True)
df.head()
```

- After merging, we were able to observe from the dataset that it does not include enough information to solve the issue statement. So, we need to figure out how much longer the lease is from this year (2023) onwards. We also need to add geocoding to the **OneMap API** so that we can figure out how far each flat is from its nearest MRT station and how far each flat is from the central business district (based on Raffles Place). Before beginning the geocoding process, any missing values and/or duplicate entries in the raw dataset would be eliminated. Additionally, to access the flat's geo-location, the address of each flat must be generated.

```
df['address'] = df['block'] + " " + df['street_name']
address_list = df['address'].unique()
```

- The picture below shows how geocoding is done using JSON requests to query and get the geolocation of each flat and each MRT station. With these, we will be able to calculate the **distance between each apartment and the MRT station** that is located closest to it. In addition, given the geolocations of all the apartments, we can calculate

the **distance that separates each flat from the Central Business District**. The following is a template for writing code that will receive data from the OneMap API. It also includes a list of MRT stations that served as a basis for retrieving the geolocations of those stations.

```
query_address = list_of_mrt[i]
query_string = 'https://developers.onemap.sg/commonapi/search?searchVal='+str(query_address)+'&returnGeom=Y&getAddrDetails=Y'
resp = requests.get(query_string)

data_mrt=json.loads(resp.content)
```

- Calculating the **remaining time on the lease** for each apartment starting this year is the last column that must be added to our dataset. Because flat leases in Singapore are for a period of 99 years, a new variable must be established like the following when the commencing lease date variable is utilized: This is done so that the remaining lease can be calculated.

```
df_new['resale_price'] = df_new['resale_price'].astype('float')
df_new['floor_area_sqm'] = df_new['floor_area_sqm'].astype('float')
df_new['lease_commence_date'] = df_new['lease_commence_date'].astype('int64')
df_new['lease_remain_years'] = 99 - (2023 - df_new['lease_commence_date'])
```

- To properly manage the newly produced dataset, one of the first steps that must be taken is to verify that each of the variables is represented by the appropriate data type. The floor-level variable is one of the aspects that need to be looked at. There are a variety of floor-level ranges to choose from since the floor-level variable is a categorical variable. As a result, the **median is used in the process of mapping the floor level of each floor level**.

storey_range
06 TO 10
01 TO 05
06 TO 10
01 TO 05
06 TO 10

```
import statistics

def get_median(x):
    split_list = x.split(' TO ')
    float_list = [float(i) for i in split_list]
    median = statistics.median(float_list)
    return median

df['storey_median'] = df['storey_range'].apply(lambda x: get_median(x))
df
```

- It is now time to **extract the appropriate variables** that provide an answer to the issu e statement as our new data frame, which is employed for the construction of the decision tree regression model.

```
#cbd_dist = CBD distance
#min_dist_mrt = Distance to the nearest MRT station
#floor_area_sqm = Flat size
#lease_remain_years = Remaing years of lease
#storey_median = Floor level
#resale_price = Selling price (dependent variable)

scope_df = df[['cbd_dist','min_dist_mrt','floor_area_sqm','lease_remain_years','storey_median','resale_price']]
scope_df
```

- Now after extracting the relevant variables, we check those **variables for skewness** in data. If the data is found to be skewed, we use either **Logarithmic Transformation** or **Square Root Transformation** to handle that skewness.

```
import seaborn as sns
import matplotlib.pyplot as plt

# List of columns(continuous variables) for finding skewness
col = ['cbd_dist','min_dist_mrt','floor_area_sqm','lease_remain_years','storey_median','resale_price']

for i in col:
    plt.figure(figsize=(8, 6))
    sns.boxplot(data=df, x=i)
    plt.title(f'Boxplot of {i}')
    plt.xlabel(i)
    plt.show()
```

```
# Apply a logarithmic transformation to the required columns only.
# One need to apply it and check, in some cases it will handle the skewness, and in other cases it might not have a great
                                                    # effect on the data, so no need to apply for that columns

df1['floor_area_sqm'] = np.log(df1['floor_area_sqm'])
sns.boxplot(x='floor_area_sqm', data=df1)
plt.show()

df1['storey_median'] = np.log(df1['storey_median'])
sns.boxplot(x='storey_median', data=df1)
plt.show()

df1['resale_price'] = np.log(df1['resale_price'])
sns.boxplot(x='resale_price', data=df1)
plt.show()
```

Correlation Matrix:

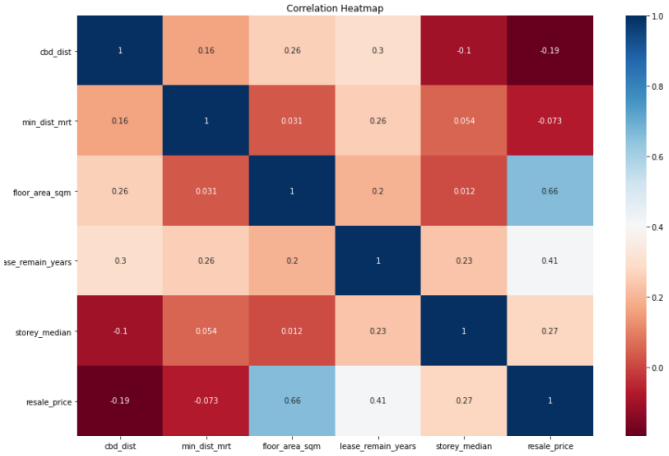

Figure 1: Correlation Matrix Resale Price

Based on the analysis, it can be deduced that the **floor size exhibits the most significant correlation with the resale price**, since this correlation is somewhat positive. The distance to the closest MRT station has the weakest negative correlation with the resale price, indicating that it has the lowest influence. Irrespective of the nature of the association, whether positive or negative, the remaining factors that influence the resale price exhibit a modest level of strength.

Decision Tree Regression:

- To increase the model accuracy, we first **normalize the data** using Standard Scaler.

```
from sklearn.preprocessing import StandardScaler

X=df1[['cbd_dist','min_dist_mrt','floor_area_sqm','lease_remain_years','storey_median']]
y=df1['resale_price']

# Normalizing the encoded data
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

- The latest dataset will be split into 90% training dataset and 10% test dataset. The dependent variable (y) is the **resale price** variable while the others are independent variables (X).

```
from sklearn.model_selection import train_test_split

# test and train split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
```

- Now, it's time to build the decision tree regression model. I have also done **hyperparameter tuning** and used **cross-validation techniques** to further enhance the accuracy of our machine learning model and decrease overfitting.

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import GridSearchCV

# Decision Tree Regressor
dtr = DecisionTreeRegressor()

# hyperparameters
param_grid = {
    'max_depth': [2, 5, 10, 15, 20, 22],
    'min_samples_split': [2, 3, 4, 5],
    'min_samples_leaf': [2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20],
    'max_features': ['auto', 'sqrt', 'log2']
}

# gridsearchcv
grid_search = GridSearchCV(estimator=dtr, param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)
print("Best hyperparameters:", grid_search.best_params_)
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
```

```
Best hyperparameters: {'max_depth': 20, 'max_features': 'auto', 'min_samples_leaf': 15, 'min_samples_split': 4}
```

Figure 2: Best Hyperparameters for our Decision Tree Regressor

- Now, to check the accuracy of our machine learning model, I have used various evaluation metrics: mean squared error (MSE), both the mean absolute error (MAE) and the root mean squared error (RMSE), and an **R-squared score** of **0.8725**, which is quite good.

```
# evalution metrics
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
print(" ")
print('Mean squared error:', mse)
print('Mean Absolute Error', mae)
print('Root Mean squared error:', rmse)
print(" ")
print('R-squared:', r2)
```

```
Mean squared error: 0.013992937995985295
Mean Absolute Error 0.09316986944583869
Root Mean squared error: 0.1182917494839995

R-squared: 0.8725657279916519
```

Figure 3: MSE, MAE, RMSE and R2 Score

Conclusion:

In conclusion, it can be said that the explanatory factors exhibit a statistically significant correlation with the selling price of a Housing and Development Board (HDB) unit. This enables us to elucidate the way each explanatory variable influences the fluctuations in the selling price of a Housing and Development Board (HDB) resale apartment. Moreover, in comparison to the other explanatory factors, the floor size has the strongest correlation with the resale price of an HDB apartment, displaying a positive and moderate association. To enhance the study, it would be beneficial to take into account elements such as the specific planning regions in which the HDB flats are situated, as well as the political borders that include these HDB flats. This consideration will help address the issue statement more comprehensively.