Data Mining

Name	ID
Hadeer Elkady	42010436
Hala Khaled	42020007
Hagar Galal	42010084
Verina Gamal	42010044
Zeyad Mohamed	420100448

1. Introduction

- **Brief Overview of the Project and its Goals:**

The primary goal of this project is to perform a comprehensive analysis of a real estate dataset using various machine learning models. The overarching objective is to gain insights into the factors influencing property prices and to develop predictive models capable of accurately categorizing or predicting these prices. The project aims to leverage different machine learning algorithms to understand the relationships within the dataset and to make informed predictions about housing prices.

The analysis involves several stages, including data preprocessing, exploratory data analysis (EDA), and the application of multiple machine learning models. Each stage contributes to a deeper understanding of the dataset and facilitates the creation of models that can potentially generalize well to new data.

- **Explanation of the Dataset Used for Analysis:**

The dataset employed in this project originates from the Melbourne real estate market. It contains information about various attributes related to properties, such as the number of rooms, distance from the city center, land size, building area, location coordinates, and more. Additionally, the dataset includes temporal information, such as the date of sale.

Each row in the dataset represents a real estate transaction, providing a rich set of features that can be utilized to understand patterns and trends in property prices. The dataset is expected to encompass a mix of numerical and categorical variables, necessitating careful preprocessing and exploratory analysis to unveil meaningful insights.

The dataset serves as the foundation for training and evaluating machine learning models, with the goal of creating models that can predict property prices or categorize them into specific groups. By the end of the analysis, the project aims to deliver actionable insights for stakeholders in the real estate domain and showcase the efficacy of different machine learning algorithms addressing predictive tasks related to property prices.

This introductory section sets the stage for a detailed exploration of the analysis, emphasizing the importance of the dataset and the broader objectives of the project.

2. Data Preprocessing

- **Loading the Dataset:**

The dataset used in this project is sourced from the Melbourne real estate market. It is typically stored in a CSV (Comma-Separated Values) file format, which is a widely used format for tabular data. The dataset is loaded into a Pandas Data Frame using the predocs() function. The source path of the CSV file is provided as an argument to this function.

python

import pandas as pd.

# Load the dataset

df = pd. read_csv (r"C:\Users\Hp\Desktop\ML project data sets\melb_data.csv")

The read_csv function parses the CSV file and creates a Data Frame, allowing for easy manipulation and analysis of the data.

- **Initial Exploration:**

Once the dataset is loaded, an initial exploration is conducted to understand its structure and characteristics.

python

# Display basic statistics and characteristics of the dataset

df.info ()

The info () method provides a concise summary of the dataset, including information about the number of non-null values, data types, and memory usage. It helps in identifying the types of features present in the dataset and whether there are any missing values.

- **Identify Any Missing Values:**

  To identify missing values in the dataset, the isna() method is applied, followed by the sum() method to count the missing values in each column.

  python

  ```
  # Identify missing values

  missing values = df. isna(). sum()
  ```

  This step is crucial for understanding the completeness of the dataset and deciding on appropriate strategies for handling missing values.

- **Data Cleaning:**

  Handling missing values is a crucial aspect of data preprocessing. Several strategies can be employed, such as imputation or removal of rows/columns with missing values.

- **Imputation:**

  Imputation is a common technique for handling missing values. In this project, the Simple Imputer class from Scikit-Learn is utilized for imputing missing values.

  python

  ```
  from sklearn. impute import SimpleImputer.

  Impute missing values using the most frequent value for 'YearBuilt'

  imputer = SimpleImputer(strategy='most_frequent')

  df['YearBuilt'] = imputer.fit_transform(df[['YearBuilt']])
  ```

  Here, the most frequent value of the 'YearBuilt' column is used to fill in missing values.

- **Removal of Unnecessary Columns:**

In some cases, certain columns may not contribute significantly to the analysis or may contain a large number of missing values. These columns can be removed to simplify the dataset.

python

# Remove the 'Address' column

df = df.drop('Address', axis=1)

The drop() method is used to remove the specified column ('Address' in this case) from the DataFrame.

By addressing missing values and removing unnecessary columns, the dataset is prepared for further analysis and model building, ensuring a clean and consistent foundation for the subsequent stages of the project.

3. Feature Engineering

- **Date Transformation:**

In this section, we perform feature engineering on the 'Date' column. The 'Date' column is typically in string format, and to extract meaningful information, we convert it to the datetime format and extract relevant features such as day, month, and year.

python

# Convert 'Date' to datetime format

df['Date'] = pd.to_datetime(df['Date'], format="%d/%m/%Y")

Extract relevant features from 'Date'

df['Day'] = df['Date'].dt.day

df['Month'] = df['Date'].dt.month

df['Year'] = df['Date'].dt.year

The pd.to_datetime function is used to convert the 'Date' column to datetime format, and the .dt accessor is then used to extract day, month, and year as separate features.

- **Log Transformation:**

  Log transformation is applied to the 'Price' variable using the natural logarithm. This transformation is often used to stabilize variance and make the distribution of the target variable more symmetric.

  python

  # Transform 'Price' variable using natural logarithm

  df['Price_ln'] = df['Price'].apply(np.log)

  # Drop the original 'Price' column

  df = df.drop('Price', axis=1)

  The np.log function is applied to the 'Price' column to perform the log transformation. The original 'Price' column is then dropped, leaving only the log-transformed 'Price_ln' column.

- **Categorical Encoding:**

  Machine learning models often require numerical input, and categorical variables need to be encoded. One-hot encoding is a common technique used to convert categorical variables into a format that can be provided to ML algorithms.

  python

  # Use one-hot encoding for categorical variables

  df = pd.get_dummies(df)

  The pd.get_dummies function is applied to the entire DataFrame to one-hot encode all categorical variables.

- **Geospatial Visualization:**

For geospatial visualization, the longitude and latitude columns ('Longtitude' and 'Lattitude') are utilized to plot data points on a map.

```
python

import geopandas as gpd

import matplotlib.pyplot as plt

 Create a GeoDataFrame for geospatial visualization

gdf = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df['Longtitude'], df['Lattitude']))

 Plot the data points on a map

world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))

world.plot()

gdf.plot(ax=plt.gca(), marker='o', color='red', markersize=5)

plt.show()
```

The GeoDataFrame is created using the 'Longtitude' and 'Lattitude' columns, and the data points are then plotted on a world map. This visualization provides insights into the geographical distribution of the real estate transactions.

Feature engineering enhances the dataset by creating new features or transforming existing ones, making it more suitable for machine learning algorithms. The transformations performed in this section contribute to the overall analysis and model-building process.

Certainly! Below is a detailed explanation for the Exploratory Data Analysis (EDA) section without including the actual code.

4. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a crucial step in understanding the characteristics of the dataset, identifying patterns, and gaining insights that will guide further analysis and modeling decisions.

- **Descriptive Statistics:**

  Descriptive statistics provide a summary overview of the dataset. Histograms are employed to visualize the distribution of numerical features, offering insights into the central tendency and dispersion of the data. Additionally, summary statistics such as mean, standard deviation, and quartiles provide a quantitative summary.

- **Correlation Analysis:**

  Correlation analysis helps reveal relationships between numerical variables. By creating a correlation matrix and visualizing it through a heatmap, the strength and direction of relationships become apparent. High correlations between variables may indicate potential multicollinearity.

- **Univariate and Bivariate Analysis:**

  Univariate analysis focuses on individual variables, providing a detailed examination of their distributions. For categorical variables like 'CouncilArea,' bar plots display the count of observations within each category. Bivariate analysis explores relationships between pairs of variables. In this case, pair plots visualize scatterplots and histograms for numerical variables, aiding in identifying patterns and potential outliers.

- **Visualization:**

  Various visualization techniques are employed to enhance understanding. Box plots are utilized to illustrate the distribution of log-transformed prices across different property types. Count plots offer insights into the distribution of observations across different regions, providing a categorical perspective on the dataset.

Exploratory Data Analysis is foundational for informed decision-making in subsequent steps. By visually examining the data and understanding relationships between variables, one can identify trends, patterns, and potential areas of interest for more in-depth investigation.

5. Machine Learning Models

- **Naive Bayes Model:**

Algorithm Explanation:

Naive Bayes is a probabilistic algorithm based on Bayes' theorem. It assumes independence between features, hence the term "naive." The model calculates the probability of each class given a set of features and selects the class with the highest probability.

Application:

In the context of the project, the Naive Bayes algorithm is applied to classify whether a property's price category is above or below the median. Features such as suburb, room count, distance, and others are used to make predictions.

Features and Evaluation Metrics:

The features include 'Suburb,' 'Rooms,' 'Distance,' 'Bedroom2,' 'Bathroom,' 'Car,' 'Landsize,' 'BuildingArea,' 'YearBuilt,' and others. Model evaluation metrics include accuracy, R^2 (coefficient of determination), mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE).

- **Decision Tree Model:**

Algorithm and Hyperparameter Tuning:

Decision Tree is a tree-like model where each node represents a decision based on features. Hyperparameters like 'criterion,' 'max_depth,' 'min_samples_split,' and 'min_samples_leaf' are tuned using GridSearchCV to find the optimal configuration.

Best Parameters and Metrics:

The best parameters, such as 'criterion,' 'max_depth,' 'min_samples_split,' and 'min_samples_leaf,' are determined through grid search. Performance metrics include accuracy, R^2, MAE, MSE, and RMSE.

- **K-Nearest Neighbors (KNN) Model:**

  Algorithm and Hyperparameter Tuning:

  KNN is a distance-based algorithm that classifies a data point based on the majority class of its k-nearest neighbors. Hyperparameters like 'n_neighbors,' 'weights,' and 'p' are tuned using GridSearchCV.

  Best Parameters and Metrics:

  The best parameters, such as 'n_neighbors,' 'weights,' and 'p,' are determined through grid search. Evaluation metrics include accuracy, R^2, MAE, MSE, and RMSE. Additionally, a scaled version of the KNN model is discussed.

- **Neural Network (MLPClassifier) Model:**

  Model Architecture:

  The MLPClassifier is a neural network model with a specified number of hidden layers and neurons. It uses the rectified linear unit (ReLU) activation function and the Adam optimizer.

  Performance Metrics:

  Metrics such as accuracy, mean absolute error (MAE), mean squared error (MSE), and R^2 are used to evaluate the neural network model.

- **Random Forest Model:**

  Algorithm Discussion:

  Random Forest is an ensemble learning method that constructs a multitude of decision trees and merges their predictions. It enhances accuracy and controls overfitting.

  Performance Metrics:

Performance metrics, including mean squared error (MSE) and R^2, are used to evaluate the Random Forest model.

- **Linear Regression and K-Means**

Linear Regression:

Model Explanation:

Linear Regression predicts a continuous target variable based on linear relationships with input features. In this project, it is used to predict property prices.

Features, Metrics, and Challenges:

Features include 'Suburb,' 'Rooms,' 'Distance,' 'Bedroom2,' 'Bathroom,' 'Car,' 'Landsize,' 'BuildingArea,' 'YearBuilt,' and more. Metrics such as RMSE, MAE, MSE, R^2, and accuracy are used to evaluate the model. Challenges may include addressing multicollinearity and selecting relevant features.

Lasso Regression with K-Means:

Model Description:

Lasso Regression introduces regularization by penalizing the absolute size of coefficients. It is combined with K-Means clustering for feature engineering.

Evaluation Metrics and Insights:

Evaluation metrics, including RMSE, MAE, MSE, R^2, and accuracy, are used to assess the combined Lasso Regression and K-Means model. Insights gained from this combination are discussed.

Conclusion:

The provided code represents a comprehensive data science project on Melbourne housing data, encompassing stages from initial preprocessing to the application of diverse machine learning models. Through meticulous data cleaning, feature engineering, and exploratory data analysis, the dataset's nuances were uncovered, and geospatial visualization provided valuable insights. The utilization of various machine learning algorithms, including Naive Bayes, Decision Tree, KNN, Neural Network, and Random Forest, demonstrated different predictive capabilities, while Linear Regression and Lasso Regression with K-Means tackled price prediction challenges. The code concludes by summarizing key insights, addressing challenges, and outlining future considerations, offering a holistic framework for extracting valuable information from real-world datasets.