**Project Report**

|  |  |
| --- | --- |
| **Name** | KRUNALSINH CHHASATIYA |
| **College** | PARUL INSTITUTE F TECHNOLOGY |
| **Program Name** | DATA SCIENCE |

**Project - 1**

**Name of the Project:** Scrapping of a Website called JUMIA([www.jumia.co.ke](http://www.jumia.co.ke))

**Tools Used:**

**1. Programming Language**

* **Python**: The project is implemented in Python, a popular language for web scraping due to its simplicity and rich ecosystem of libraries.

**2. Libraries/Frameworks**

* **requests**: Used to send HTTP requests to the JUMIA website and retrieve the HTML content of the web pages.
* **BeautifulSoup** (from bs4): A library for parsing and navigating HTML or XML documents. It extracts structured data, such as product names, prices, and ratings.
* **pandas**: A data manipulation library used to organize the extracted data into a tabular format (e.g., Data Frames) and save it in formats like CSV.

**3. Tools/Platforms**

* **Jupyter Notebook**: The project is structured as a .ipynb notebook, providing an interactive environment for coding and testing.
* **Web Browser** (indirectly): Used to inspect the JUMIA website's HTML structure (via developer tools) to identify the relevant tags and classes for data extraction.

**4. Data Storage**

* **CSV Files**: The project likely saves the scraped data in CSV format for easy access and analysis.

**Optional Enhancements (if applicable):**

* **html.parser**: Parsing engines used by BeautifulSoup.
* **Error Handling**: Try-except blocks to handle potential issues like missing data or connectivity errors.

**Working Procedure:**

 **Step 1: Import Libraries:**

* Import essential Python libraries: requests for fetching web pages, BeautifulSoup for parsing HTML, and pandas for organizing data.

 **Step 2: Analyze the Website:**

* Use the browser's developer tools to inspect the JUMIA website's HTML structure.
* Identify the relevant tags and classes containing product information, such as names, prices, and ratings.

 **Step 3: Fetch Web Pages:**

* Use the requests.get() method to send HTTP GET requests to the target URL.
* Append a query parameter to navigate through multiple pages.

 **Step 4: Parse HTML Content:**

* Parse the retrieved HTML content using BeautifulSoup and locate the desired elements using methods like find\_all().

 **Step 5: Extract Data:**

* Loop through the HTML elements to extract product details (name, price, and rating) and store them in separate lists.

 **Step 6: Handle Missing Data:**

* Use error handling (e.g., try-except) to manage missing or inconsistent data fields.

 **Step 7: Store Data:**

* Organize the extracted data into a pandas DataFrame.
* Save the DataFrame as a CSV file for analysis or future use.

 **Step 8: Validate Results:**

* Verify the output to ensure accuracy and completeness of the scraped data.

**Learning Outcomes:**

* **Web Scraping Techniques**:
  + Gained practical experience in extracting data from websites using Python libraries like requests and BeautifulSoup.
* **HTML Structure Understanding**:
  + Learned to inspect and interpret website HTML to locate specific elements for data extraction.
* **Data Handling**:
  + Enhanced skills in organizing and cleaning scraped data using pandas.
* **Error Handling**:
  + Learned to manage potential issues, such as missing data or connectivity errors, with robust error-handling techniques.
* **Pagination Logic**:
  + Understood how to scrape data from multiple pages by dynamically constructing URLs.
* **CSV Storage**:
  + Developed the ability to save and organize data in CSV format for future use and analysis.

**Project - 2**

**Name of the Project:** Classification of Brazil Forest Fires Dataset using Pandas.

**Tools Used:**

1. **Programming Language**

* **Python**: The project is implemented in Python, a widely used language for data analysis and machine learning.

2. **Libraries/Frameworks**

* **pandas**: Used for data manipulation, cleaning, and exploratory data analysis (EDA).
* **numpy**: Utilized for numerical operations and handling arrays.
* **matplotlib and seaborn**: Employed for visualizing data through plots, charts, and heatmaps.
* **scikit-learn**: Used for machine learning tasks, such as splitting data, building classification models, and evaluating their performance.
* Models like Decision Trees, Random Forest, or Logistic Regression may have been used for classification.

3. **Tools/Platforms**

* **Jupyter Notebook**: The project is organized and executed within a .ipynb notebook, providing an interactive coding environment.
* **Python IDEs** (optional): Tools like PyCharm or VS Code may also be used for development.

4. **Dataset**

* **Brazil Forest Fires Dataset**: A dataset containing features related to forest fires in Brazil, used as input for classification tasks.

**Working Procedure:**

 **Step 1: Import Libraries:**

* Load necessary Python libraries like pandas, numpy, matplotlib, seaborn, and scikit-learn.

 **Step 2: Load Dataset:**

* Load the Brazil Forest Fires dataset into a pandas DataFrame.
* Inspect the data using methods like .head(), .info(), and .describe() to understand its structure and features.

 **Step 3: Data Cleaning:**

* Handle missing values by filling, imputing, or removing them.
* Convert categorical features into numerical representations using techniques like one-hot encoding or label encoding.
* Normalize or scale numerical data for better model performance.

 **Step 4: Exploratory Data Analysis (EDA):**

* Use matplotlib and seaborn to visualize data distributions, correlations, and relationships between features.
* Identify key patterns and outliers that may affect the classification task.

 **Step 5: Data Splitting:**

* Divide the dataset into training and testing sets using train\_test\_split() from scikit-learn.

 **Step 6: Model Building:**

* Choose classification models like Decision Trees, Random Forest, Logistic Regression.
* Train models on the training dataset and tune hyper-parameters if necessary.

 **Step 7: Evaluation:**

* Evaluate model performance using metrics like accuracy, precision, recall, F1-score, and confusion matrix.

 **Step 8: Interpretation:**

* Analyze results to determine the best-performing model and draw insights from the data.

**Learning Outcomes:**

* **Data Preprocessing Skills**:
  + Gained experience in cleaning, handling missing values, and encoding categorical data for machine learning tasks.
* **Exploratory Data Analysis (EDA)**:
  + Learned to visualize data distributions and uncover patterns using matplotlib and seaborn.
* **Model Building and Evaluation**:
  + Developed skills in building classification models (e.g., Decision Trees, Random Forest) and evaluating them using metrics like accuracy and F1-score.
* **Feature Engineering**:
  + Understood the importance of feature selection, scaling, and transformation for improving model performance.
* **Python Proficiency**:
  + Enhanced proficiency in Python libraries like pandas, numpy, and scikit-learn for data analysis and machine learning.
* **Real-World Problem Solving**:
  + Applied theoretical concepts to a practical problem, gaining insights into forest fire classification.

**Project - 3**

**Name of the Project: Toxic Comment Classification.**

**Tools Used:**

**1. Programming Language**

* **Python**: The project is implemented in Python, a popular language for web scraping due to its simplicity and rich ecosystem of libraries.

### 2. Libraries and Frameworks

* **NumPy**: For numerical computations (import numpy as np).
* **Pandas**: For data manipulation and analysis (import pandas as pd).
* **Sklearn** (Scikit-learn):
  + **Train-test split**: For splitting the dataset (train\_test\_split).
  + **TfidfVectorizer**: For text vectorization.
  + **PassiveAggressiveClassifier**: For classification.

**3. Tools/Platforms**

* **Jupyter Notebook**: The project is structured as a .ipynb notebook, providing an interactive environment for coding and testing.

**Working Procedure:**

 **Data Loading**:

* Load the dataset into a pandas DataFrame for analysis and manipulation.

 **Data Exploration**:

* Perform an initial inspection of the dataset.
* Check for labels and textual data to understand its structure and contents.

 **Data Splitting**:

* Split the dataset into training and testing sets using train\_test\_split from Scikit-learn.
* Typically, 80% of the data is used for training and 20% for testing.

 **Text Preprocessing**:

* Utilize TfidfVectorizer to transform textual data into numerical features.
* Remove stop words and calculate term frequency-inverse document frequency (TF-IDF) values.

 **Model Initialization and Training**:

* Initialize the PassiveAggressiveClassifier with appropriate hyperparameters (e.g., max\_iter).
* Train the model using the TF-IDF-transformed training data.

 **Prediction**:

* Use the trained model to predict labels for the test dataset.

 **Evaluation**:

* Evaluate the model's performance using metrics like accuracy and confusion matrix.

 **Result Analysis**:

* Analyze the classification results and metrics to assess the model's effectiveness in detecting toxic comments.

**Learning Outcomes:**

 **Data Preprocessing**:

* Learned the importance of text preprocessing techniques like TF-IDF for converting text into numerical features.

 **Model Training**:

* Gained experience in using machine learning models, specifically the PassiveAggressiveClassifier, for classification tasks.

 **Evaluation Metrics**:

* Understood how to evaluate model performance using metrics like accuracy and confusion matrix.

 **Scikit-learn Usage**:

* Enhanced proficiency in Scikit-learn libraries for splitting data, transforming features, and training models.

 **Practical Problem-Solving**:

* Developed skills to classify text data effectively for a real-world application like toxic comment detection.

**Project - 4**

**Name of the Project:** Cab Fare Price Prediction.

**Tools Used:**

**1. Programming Language**

* **Python**: The project is implemented in Python, a popular language for web scraping due to its simplicity and rich ecosystem of libraries.

### 2. Libraries and Frameworks

* **Operating System Utilities**:
  + os: For interacting with the file system.
* **Data Manipulation**:
  + Pandas: For handling and processing datasets (import pandas as pd).
  + NumPy: For numerical operations (import numpy as np).
* **Data Visualization**:
  + Matplotlib: For plotting data (import matplotlib.pyplot as plt).
  + Seaborn: For advanced data visualizations (import seaborn as sns).
* **Data Analysis and Modeling**:
  + **Scikit-learn**:
    - Models: DecisionTreeRegressor, RandomForestRegressor, GradientBoostingRegressor, LinearRegression.
    - Utilities: train\_test\_split for data splitting, mean\_squared\_error and r2\_score for evaluation metrics.
    - Hyperparameter Tuning: GridSearchCV for optimizing model parameters.

**3. Tools/Platforms**

* **Jupyter Notebook**: The project is structured as a .ipynb notebook, providing an interactive environment for coding and testing.

**Working Procedure:**

 **Problem Understanding**:

* Define the problem as predicting cab fare prices based on historical ride data.
* Identify key factors influencing fare, such as distance, time, and location.

 **Data Loading**:

* Load the dataset using pandas for analysis and manipulation.

 **Data Exploration**:

* Inspect the dataset to understand its structure and identify relevant features.
* Perform exploratory data analysis (EDA) using matplotlib and seaborn to visualize trends and relationships.

 **Data Cleaning**:

* Handle missing or inconsistent values.
* Remove outliers and address potential data quality issues.

 **Feature Engineering**:

* Create new features, such as distance between pickup and drop-off points.
* Transform categorical features (if any) into numerical representations.

 **Data Splitting**:

* Split the dataset into training and testing sets using train\_test\_split.

 **Model Selection and Training**:

* Train multiple regression models, including LinearRegression, DecisionTreeRegressor, RandomForestRegressor, and GradientBoostingRegressor.

 **Hyperparameter Tuning**:

* Use GridSearchCV to optimize model parameters for better performance.

 **Model Evaluation**:

* Evaluate models using metrics like mean\_squared\_error and r2\_score.

 **Result Analysis**:

* Compare model performances and select the best model for prediction.

**Learning Outcomes:**

 **Data Handling**:

* Gained expertise in loading, exploring, and cleaning real-world datasets using pandas.

 **Exploratory Data Analysis (EDA)**:

* Learned to visualize and interpret data trends using matplotlib and seaborn.

 **Feature Engineering**:

* Developed skills to create meaningful features, such as distance calculations and time-based variables.

 **Model Building**:

* Acquired knowledge of regression models like LinearRegression, RandomForestRegressor, and GradientBoostingRegressor.

 **Hyper-parameter Tuning**:

* Gained experience in optimizing model performance using GridSearchCV.

 **Evaluation Techniques**:

* Understood evaluation metrics like mean\_squared\_error and r2\_score to assess model performance.

 **Practical Insights**:

* Learned how machine learning can be applied to predict cab fares accurately in a real-world scenario.

**Project - 5**

**Name of the Project:** Customer Transaction Analysis.

**Tools Used:**

**1. Programming Language**

* **Python**: The project is implemented in Python, a popular language for web scraping due to its simplicity and rich ecosystem of libraries.

**2.Libraries/Frameworks**

 **os**: Used for operating system interactions.

 **numpy**: A library for numerical computations and array manipulations.

 **pandas**: A powerful library for data manipulation and analysis.

 **seaborn**: A library for statistical data visualization.

 **matplotlib.pyplot**: Used for creating static, animated, and interactive visualizations.

 **lightgbm**: A gradient boosting framework for machine learning tasks.

 **scikit-learn**:

* **model\_selection**: For splitting datasets and cross-validation.
  + Functions: train\_test\_split, StratifiedKFold
* **ensemble**: For implementing ensemble methods.
  + Models: RandomForestClassifier, RandomForestRegressor
* **linear\_model**: For linear and logistic regression models.
  + Model: LogisticRegression
* **metrics**: For model evaluation metrics.
  + Metrics: confusion\_matrix, roc\_curve, permutation\_importance

 **pdpbox**: A library for creating partial dependence plots, often used for model interpretability.

 **imblearn**: Used for handling imbalanced datasets.

* Technique: SMOTE (Synthetic Minority Over-sampling Technique)

 **warnings**: Used to manage and filter warning messages.

**3. Tools/Platforms**

* **Jupyter Notebook**: The project is structured as a .ipynb notebook, providing an interactive environment for coding and testing.

**Working Procedure:**

 **Data Collection**:  
Gathered transactional data from a reliable source to analyze customer behavior and trends.

 **Data Preprocessing**:

* Handled missing values and outliers to ensure data quality.
* Converted categorical variables into numerical representations using encoding techniques (e.g., one-hot encoding).
* Scaled numerical features for uniformity.

 **Exploratory Data Analysis (EDA)**:

* Used libraries like Pandas, Seaborn, and Matplotlib to visualize patterns and distributions in the data.
* Identified key insights, such as high-value customers, common transaction types, and seasonal trends.

 **Feature Engineering**:

* Created new variables like average transaction value, customer segmentation, and frequency of purchases.
* Used domain knowledge to enhance the dataset with meaningful features.

 **Model Building**:

* Split the dataset into training and testing sets using train\_test\_split.
* Built and tuned machine learning models, including Random Forest and LightGBM, for classification or regression tasks.

 **Model Evaluation**:

* Assessed model performance using metrics like accuracy, confusion matrix, ROC curve, and feature importance analysis.

 **Model Interpretability**:

* Used tools like PDPBox and SHAP to understand the model’s decision-making process.

 **Handling Imbalanced Data**:

* Applied SMOTE to balance the dataset and improve model performance.

 **Insights and Reporting**:  
Summarized key findings, actionable insights, and recommendations for business decisions.

**Learning Outcomes:**

* Gained expertise in handling and preprocessing large datasets for analysis.
* Learned to use visualization tools (e.g., Matplotlib, Seaborn) for exploring data trends and patterns.
* Developed proficiency in feature engineering to enhance predictive model performance.
* Acquired skills in implementing machine learning models like Random Forest and LightGBM for classification and regression tasks.
* Mastered techniques for addressing data imbalances using SMOTE.
* Understood the importance of model evaluation metrics (e.g., ROC curve, confusion matrix) for assessing performance.
* Learned to interpret machine learning models using tools like PDPBox for explainability.
* Improved ability to derive actionable insights from data analysis for business decisions.