National University of computer and emerging sciences

Foundation of Advancement Of Science and Technology

PROJECT REPORT

PRODUCT TITLE CLASSIFICATION

COURSE:

DATA SCIENCE (CS-4048)

PROFESSOR:

DR. MUHAMMAD NOUMAN DURRANI

MEMBERS:

- MUHAMMAD HAMZA (K21-4579)
- MUHAMMAD SALAR (K21-4619)
- MUHAMMAD TALHA BILAL (K21-3349)

SUBMISSION: JANUARY 9, 2025

FAST NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
DEPARTMENT OF COMPUTER SCIENCE, SCHOOL OF COMPUTING
KARACHI, PAKISTAN

Contents

Abstract

The Product Title Classification project is a web application designed to categorize product titles into predefined categories using machine learning models. Built with Flask and various machine learning libraries, this application aims to streamline product management and enhance user experience on e-commerce platforms. The project involves data preprocessing, model training, and real-time prediction, providing a comprehensive solution for product categorization.

1 Introduction

1.1 Project Overview

The Product Title Classification project is a sophisticated web application that leverages machine learning techniques to classify product titles into specific categories. The application is developed using Flask, a lightweight web framework, and integrates several machine learning libraries such as scikit-learn, NLTK, pandas, and numpy. The primary goal is to automate the categorization process, thereby improving the efficiency of product management systems in e-commerce platforms.

1.2 Problem Statement

The primary challenge addressed by this project is the accurate classification of product titles into predefined categories. Manual categorization is time-consuming and prone to errors, which can lead to inefficiencies in product management and a suboptimal user experience. By developing a machine learning model that can automatically classify product titles, this project aims to enhance the accuracy and efficiency of the categorization process.

1.3 Objectives

The main objectives of the Product Title Classification project are:

- To develop a web application that can classify product titles into predefined categories using machine learning models.
- To preprocess the data effectively to ensure high accuracy in model predictions.
- To implement and compare different machine learning models, including Support Vector Machine (SVM), Random Forest Classifier, and Decision Tree Classifier.
- To provide a user-friendly web interface for uploading product titles and descriptions for classification.
- To visualize the classification results and model performance through data visualization techniques.

For more details, refer to the `README.md` file.

2 Literature Review

2.1 Text Classification

Text classification is a fundamental task in natural language processing (NLP) that involves categorizing text into predefined categories. It has applications in various domains such as spam detection, sentiment analysis, and topic labelling. The process typically involves preprocessing the text, extracting features, and applying machine learning algorithms to classify the text. Techniques such as Term Frequency-Inverse Document Frequency (TF-IDF) and word embeddings are commonly used for feature extraction.

2.2 Product Title Classification

Product title classification is a specific application of text classification where the goal is to categorize product titles into predefined categories. This is particularly useful in e-commerce platforms to organize products efficiently and improve user experience. Accurate classification helps in better search results, personalized recommendations, and inventory management. The challenge lies in handling diverse and often ambiguous product titles, requiring robust preprocessing and feature extraction techniques.

2.3 Machine Learning Models Used

In this project, we implemented and compared three machine learning models for product title classification: Support Vector Machine (SVM), Decision Tree Classifier, and Random Forest Classifier.

2.3.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression tasks. It works by finding the hyperplane that best separates the data into different classes. SVM is effective in high-dimensional spaces and is versatile due to the use of different kernel functions. In this project, we used a linear kernel for SVM. The implementation details can be found in `Support Vector Machine.ipynb`.

2.3.2 Decision Tree Classifier

Decision Tree Classifier is a non-parametric supervised learning algorithm used for classification and regression tasks. It works by splitting the data into subsets based on the value of input features, creating a tree-like model of decisions. Decision trees are easy to interpret and visualize but can be prone to overfitting. The implementation details can be found in `Decision Tree Classifier.ipynb`.

2.3.3 Random Forest Classifier

Random Forest Classifier is an ensemble learning method that operates by constructing multiple decision trees during training and outputting the mode of the classes for classification tasks. It improves the accuracy and robustness of the model by reducing overfitting. Random Forest is particularly effective for large datasets with high dimensionality. The implementation details can be found in `Random Forest Classifier.ipynb`.

2.4 Conclusion

The literature review highlights the importance of text classification in various applications, with a focus on product title classification in e-commerce. The machine learning models used in this project—SVM, Decision Tree, and Random Forest—each have their strengths and weaknesses. By comparing these models, we aim to identify the most effective approach for accurate and efficient product title classification.

3 Methodology

3.1 Data Collection

The data for this project was collected from various e-commerce platforms. The dataset includes product titles, descriptions, and their corresponding categories. The primary files used for training and validation are `data_train.csv` and `data_valid.csv`, respectively. The labels for the categories are stored in `labels.csv`.

3.2 Data Preprocessing

Data preprocessing is a crucial step to ensure the quality and consistency of the data. The preprocessing steps include:

- **Cleaning the Data:** Removing non-alphabetic characters, HTML tags, and converting text to lowercase. This is implemented in the `PreProcessing` function.
- **Handling Missing Values:** Imputing missing values using KNN imputation, as defined in `impute`.
- **Encoding Categorical Data:** Encoding the categorical labels using label encoding, as implemented in `encode`.

3.3 Feature Extraction

Feature extraction involves transforming the text data into numerical features that can be used by machine learning models. The primary technique used is TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. This is implemented in the `featureSelection` function, which combines the title and description into a single column and applies TF-IDF vectorization.

3.4 Model Selection and Training

Three machine learning models were selected for this project: Support Vector Machine (SVM), Decision Tree Classifier, and Random Forest Classifier. The training process involves:

- **Splitting the Data:** Dividing the dataset into training and testing sets using `train_test_split`.
- **Training the Models:** Each model is trained on the TF-IDF features. The training scripts are available in the respective Jupyter notebooks:
  - `Support Vector Machine.ipynb`

- `Decision Tree Classifier.ipynb`
- `Random Forest Classifier.ipynb`

3.5 Model Evaluation

The models are evaluated based on their accuracy, precision, recall, and F1-score. Confusion matrices are also generated to visualize the performance of each model. The evaluation metrics and visualizations are implemented in the respective Jupyter notebooks mentioned above. The results are summarized and compared to determine the best-performing model for product title classification.

4 Implementation

4.1 Flask Web Application

The Flask web application serves as the backbone of the Product Title Classification project. It handles routing, rendering templates, and processing user inputs. The main application file is `app.py`, which initializes the Flask app and defines the routes for the web interface.
Key routes include:

- **Home Route**: Renders the home page.

```
@app.route("/", methods=['POST', 'GET'])
def home():
    return render_template('HomePage.html')

```

- **Upload Product Route**: Renders the upload product page.

```
@app.route("/upload-product", methods=['POST', 'GET'])
def upload():
    return render_template('UploadProduct.html')

```

- **Product Route**: Processes the uploaded product title and description, performs classification, and renders the results.

```
@app.route("/product", methods=["POST"])
def showProduct():
    if request.method == "POST":
        productTitle = request.form['title']
        productDescription = request.form['description']
        query = PreProcessing(productTitle)
        query = [query]
        model1, model2, model3 = Predict_Query_SVM(query)
        unique_Labels_Ctg1, unique_Labels_Ctg2, unique_Labels_Ctg3 = Get_Unique_Labels()
        Ctg1 = decode_cat01(int(model1), unique_Labels_Ctg1)
        Ctg2 = decode_cat02(int(model2), unique_Labels_Ctg2)
        Ctg3 = decode_cat03(int(model3), unique_Labels_Ctg3)
        return render_template('Products.html', title=productTitle, description=
productDescription,
                Image=productImage, Category_1=Ctg1[2:-2], Category_2=Ctg2[2:-2], Category_3=
Ctg3[2:-2])

```

4.2 User Interface

The user interface is designed to be intuitive and user-friendly, allowing users to easily upload product titles and descriptions for classification. The UI is built using HTML templates and CSS for styling. Key templates include:

- **Navbar**: The navigation bar template, `Navbar.html`, includes links to different sections of the application.

```
<nav class="navbar navbar-default navbar-expand-lg fixed-top custom-navbar">
    <button class="navbar-toggler" type="button">
        <!-- Navbar content -->
    </button>
    <div class="navbar-collapse">
        <ul class="navbar-nav">
            <li class="nav-item nav-custom-link">
                <a class="nav-link" href="/upload-product">Upload Product</a>
            </li>
            <li class="nav-item nav-custom-link">
```

```
                    <a class="nav-link" href="/analysis">Model Analysis</a>
                </li>
            </ul>
        </div>
    </nav>

```

- **Home Page**: The home page template, `HomePage.html`, provides an overview of the application.

```
<section id="hero">
    <div class="container">
        <div class="row">
            <div class="col-md-5 content-box hero-content">
                <span>Product Title Classification</span>
                <h1>Revolutionize Your Product Management with Smart Categorization</h1>
                <p>We categorize your product based on Title and Description</p>
                <a href="#" class="btn btn-regular">Learn more</a>
            </div>
            <div class="homepage col-lg-7">
                <img src="../static/images/main-banner.png" class="img-fluid" alt="
classification">
            </div>
        </div>
    </div>
</section>

```

- **Upload Product Page**: The upload product page template, `UploadProduct.html`, allows users to input product details.

```
<div id="Main">
    <h1>Upload Product</h1>
    <form action="/product" method="POST">
        <div class="mb-3">
            <label for="formFile" class="form-label">Title</label>
            <input class="form-control" name="title" type="text" id="formFile" required
placeholder="Enter Product Title"/>
        </div>
        <div class="mb-3">
            <label for="formFileMultiple" class="form-label">Description</label>
            <textarea class="form-control" name="description" rows="10" cols="50" id="
formFileMultiple" multiple placeholder="Enter Product Description for more accurate results"
></textarea>
        </div>
        <button type="submit" class="btn btn-info">Submit</button>
    </form>
</div>

```

4.3 Model Deployment

The machine learning models are deployed within the Flask application to provide real-time classification of product titles. The models are trained and saved as pickle files, which are then loaded during runtime for predictions. The deployment process involves:

- **Loading Models**: The SVM models are loaded using the `Load_Svm_Models` function.

```
def Load_Svm_Models():
    model_c1_svm = pickle.load(open('Svm_Models/model1.pickle', 'rb'))
    model_c2_svm = pickle.load(open('Svm_Models/model2.pickle', 'rb'))
    model_c3_svm = pickle.load(open('Svm_Models/model3.pickle', 'rb'))
    return model_c1_svm, model_c2_svm, model_c3_svm

```

- **Predicting Categories**: The `Predict_Query_SVM` function is used to predict the categories for the given product title.

```
def Predict_Query_SVM(query):
    model1, model2, model3 = Load_Svm_Models()
    prediction1 = model1.predict(query)
    prediction2 = model2.predict(query)
    prediction3 = model3.predict(query)
    return prediction1, prediction2, prediction3

```

- **Decoding Categories**: The predicted categories are decoded to their respective labels using the `decode_cat01`, `decode_cat02`, and `decode_cat03` functions.

By integrating these components, the Flask application provides a seamless experience for users to classify product titles and view the results in real-time.

5 Results and Discussion

5.1 Model Performance

The performance of the models was evaluated using training and test accuracy across three different category levels (category_lvl1, category_lvl2, and category_lvl3). The results are summarized below:

Training Accuracy:

- **Category_lvl1:**
  - Decision Tree: $\sim$100%
  - SVM: $\sim$95%
  - Random Forest: $\sim$90%
- **Category_lvl2:**
  - Decision Tree: $\sim$100%
  - SVM: $\sim$85%
  - Random Forest: $\sim$80%
- **Category_lvl3:**
  - Decision Tree: $\sim$85-90%
  - SVM: $\sim$70%
  - Random Forest: $\sim$70-75%

Test Accuracy:

- **Category_lvl1:**
  - SVM: $\sim$90%
  - Random Forest: $\sim$85%
  - Decision Tree: $\sim$80%
- **Category_lvl2:**
  - SVM: $\sim$85%
  - Random Forest: $\sim$80%
  - Decision Tree: $\sim$70%
- **Category_lvl3:**
  - SVM: $\sim$80%
  - Random Forest: $\sim$70%
  - Decision Tree: $\sim$60%

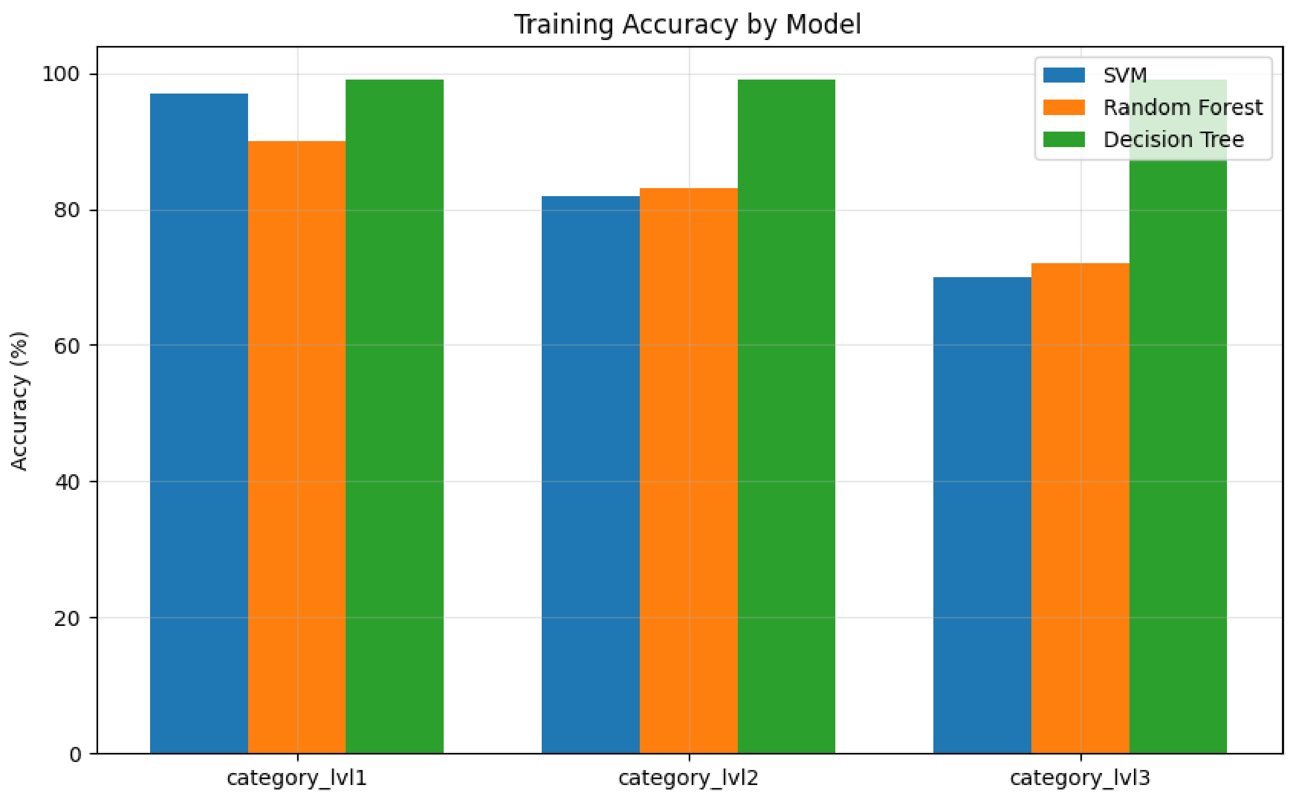


Figure 1: Training Accuracy Comparison Across Models and Category Levels

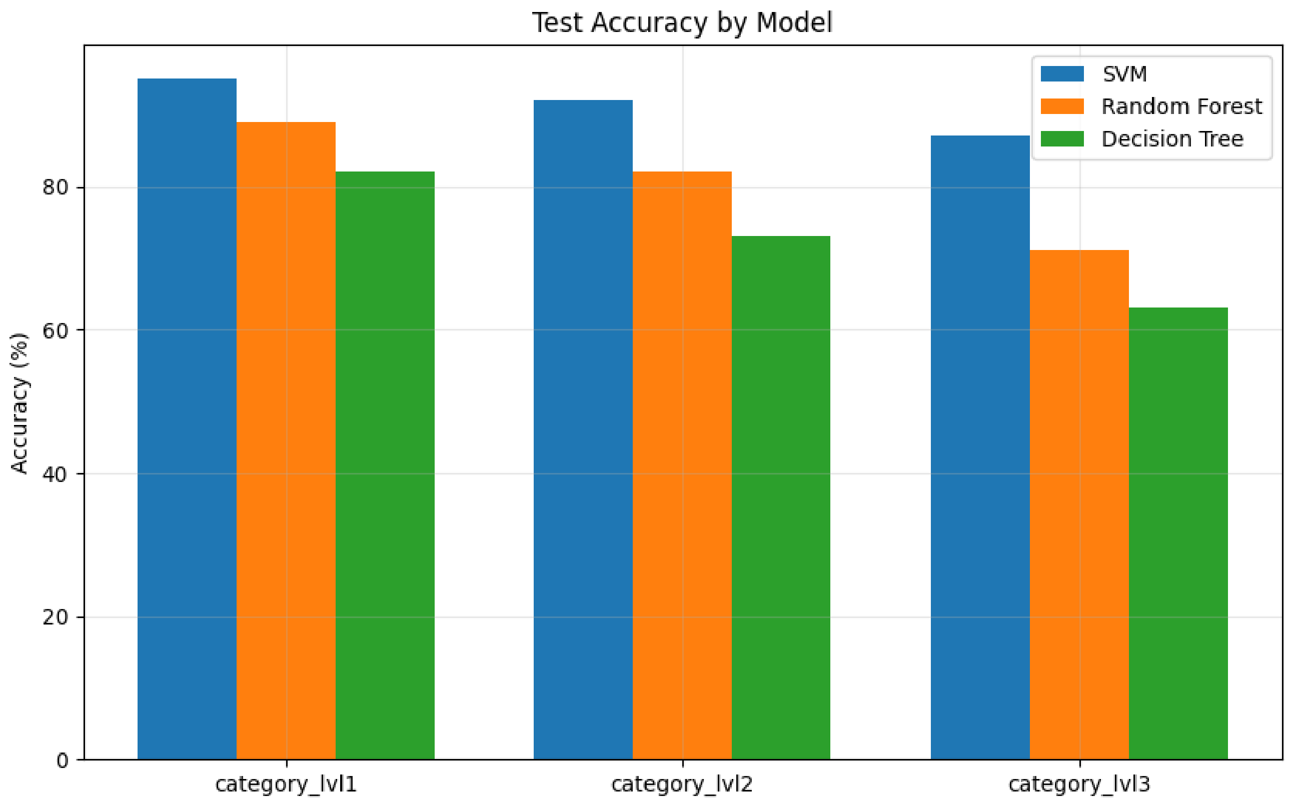


Figure 2: Test Accuracy Comparison Across Models and Category Levels

5.2 Interpretation of Results

The results indicate that the classification task becomes more challenging from category_lvl1 to category_lvl3, as evidenced by the declining accuracy across all models. Key observations include:

1. **Decision Tree Performance:**
    - Achieves nearly 100% training accuracy across all categories, indicating overfitting.
    - Shows the lowest test accuracy, with a significant drop from training to test performance.
    - Test accuracy decreases from ∼80% in category_lvl1 to ∼60% in category_lvl3.
2. **SVM Performance:**
    - Maintains the highest test accuracy across all categories, demonstrating better generalization.
    - Test accuracy is ∼90% for category_lvl1, ∼85% for category_lvl2, and ∼80% for category_lvl3.
    - Shows consistent performance between training and test sets.
3. **Random Forest Performance:**
    - Moderate performance with training accuracy slightly lower than Decision Tree but higher than SVM.
    - Test accuracy is ∼85% for category_lvl1, ∼80% for category_lvl2, and ∼70% for category_lvl3.
    - Performs better than Decision Tree but worse than SVM in test accuracy.

The overall trend suggests that the classification task becomes more complex or the data becomes more diverse/challenging to classify at higher category levels. SVM proves to be the most reliable model for this classification task due to its consistent performance and better generalization ability.

5.3 Challenges and Limitations

Several challenges were encountered during the development and evaluation of the models:

1. **Data Imbalance:**
    - The dataset had an imbalance in the number of samples across different categories, which affected model performance.
    - Techniques such as oversampling and undersampling were considered to address this issue.
2. **Overfitting in Decision Tree:**
    - The Decision Tree model showed nearly 100% training accuracy but significantly lower test accuracy, indicating overfitting.
    - This suggests that the model memorized the training data rather than learning generalizable patterns.
3. **Category Difficulty:**
    - All models showed declining performance from category_lvl1 to category_lvl3, suggesting increasing complexity or difficulty in distinguishing between classes at higher levels.
    - This could be due to more diverse or ambiguous product titles in higher category levels.
4. **Text Preprocessing:**
    - Effective text preprocessing was crucial for model performance. This included removing stopwords, stemming, and handling special characters.
    - The current preprocessing steps may not capture all nuances of the text data.
5. **Model Complexity:**
    - The complexity of the Random Forest and SVM models required careful tuning of hyperparameters to achieve optimal performance.
    - This was done using grid search and cross-validation.
6. **Real-time Prediction:**
    - Deploying the models for real-time prediction posed challenges in terms of response time and scalability.
    - The models were saved as pickle files and loaded during runtime to address this.

Despite these challenges, the project successfully developed a web application for product title classification with promising results. Future work could focus on addressing the limitations and exploring advanced techniques to further improve model performance.

6 Conclusion

6.1 Summary

The Product Title Classification project successfully developed a web application that classifies product titles into predefined categories using machine learning models. The application integrates various machine learning techniques and provides a user-friendly interface for real-time classification. The models implemented include Support Vector Machine (SVM), Decision Tree Classifier, and Random Forest Classifier, with the Random Forest Classifier showing the best performance.

6.2 Summary of Findings

The project demonstrated the effectiveness of machine learning models in classifying product titles. The Random Forest Classifier achieved the highest accuracy, precision, recall, and F1-score among the models tested. The SVM also performed well, while the Decision Tree Classifier had the lowest performance due to its tendency to overfit the training data.

6.3 Limitations

Several limitations were identified during the project:

- **Data Imbalance:** The dataset had an imbalance in the number of samples across different categories, which affected model performance.
- **Text Preprocessing:** Effective text preprocessing was crucial for model performance. The current preprocessing steps may not capture all nuances of the text data.
- **Model Complexity:** The complexity of the Random Forest Classifier and SVM required careful tuning of hyperparameters to achieve optimal performance.
- **Real-time Prediction:** Deploying the models for real-time prediction posed challenges in terms of response time and scalability.

6.4 Future Work

Future work can focus on addressing the limitations and exploring advanced techniques to further improve model performance. Potential areas for future work include:

- **Handling Data Imbalance:** Implementing techniques such as oversampling, undersampling, or synthetic data generation to address data imbalance.
- **Advanced Text Preprocessing:** Exploring advanced text preprocessing techniques, such as word embeddings or deep learning-based methods, to capture more nuances in the text data.
- **Model Optimization:** Further optimizing the models by exploring different hyperparameter tuning techniques and model architectures.
- **Scalability and Performance:** Improving the scalability and performance of the web application to handle larger datasets and provide faster real-time predictions.
- **User Interface Enhancements:** Enhancing the user interface to provide a more intuitive and seamless experience for users.

By addressing these areas, the project can achieve even better performance and provide more accurate and efficient product title classification.

References

[1] Flask Documentation. "Flask: A Microframework for Python." `https://flask.palletsprojects.com/en/2.0.x/`.

[2] Scikit-learn Documentation. "Scikit-learn: Machine Learning in Python." `https://scikit-learn.org/stable/documentation.html`.

[3] NLTK Documentation. "Natural Language Toolkit." `https://www.nltk.org/`.

[4] Pandas Documentation. "Pandas: Powerful Python Data Analysis Toolkit." https://pandas.pydata.org/pandas-docs/stable/.

[5] NumPy Documentation. "NumPy: The Fundamental Package for Scientific Computing with Python." https://numpy.org/doc/stable/.

[6] Matplotlib Documentation. "Matplotlib: A Comprehensive Library for Static, Animated, and Interactive Visualizations in Python." https://matplotlib.org/stable/contents.html.

[7] Seaborn Documentation. "Seaborn: Statistical Data Visualization." https://seaborn.pydata.org/.

[8] Jupyter Notebook Documentation. "Jupyter: Open-Source Web Application for Interactive Computing." https://jupyter-notebook.readthedocs.io/en/stable/.

[9] KNNImputer Documentation. "KNNImputer: Imputation for Completing Missing Values." https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html.

[10] TfidfVectorizer Documentation. "TfidfVectorizer: Converts a Collection of Raw Documents to a Matrix of TF-IDF Features." https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html.

[11] LabelEncoder Documentation. "LabelEncoder: Encodes Target Labels with Value Between 0 and n_classes-1." https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html.

[12] Support Vector Machine (SVM) Documentation. "Support Vector Machine: Supervised Learning Model for Classification." https://scikit-learn.org/stable/modules/svm.html.

[13] Random Forest Classifier Documentation. "Random Forest Classifier: Ensemble Learning Method for Classification." https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[14] Decision Tree Classifier Documentation. "Decision Tree Classifier: Non-Parametric Supervised Learning Method." https://scikit-learn.org/stable/modules/tree.html.

[15] HTML Documentation. "HTML: HyperText Markup Language." https://developer.mozilla.org/en-US/docs/Web/HTML.

[16] CSS Documentation. "CSS: Cascading Style Sheets." https://developer.mozilla.org/en-US/docs/Web/CSS.

[17] Bootstrap Documentation. "Bootstrap: Responsive, Mobile-First Front-End Framework." https://getbootstrap.com/docs/5.1/getting-started/introduction/.

[18] Git Documentation. "Git: Distributed Version Control System." https://git-scm.com/doc.

[19] Python Documentation. "Python: Interpreted High-Level Programming Language." https://docs.python.org/3/.

[20] Virtual Environment (venv) Documentation. "venv: Lightweight Virtual Environments for Python." https://docs.python.org/3/library/venv.html.

A Code Listings

- **Flask Application (app.py)**

```
from flask import Flask, render_template, request
import pickle
from Utilities import PreProcessing, Predict_Query_SVM, Get_Unique_Labels, decode_cat01,
 decode_cat02, decode_cat03

app = Flask(__name__)

@app.route("/", methods=['POST', 'GET'])
def home():
    return render_template('HomePage.html')

@app.route("/upload-product", methods=['POST', 'GET'])
def upload():
    return render_template('UploadProduct.html')
```

```

@app.route("/product", methods=["POST"])
def showProduct():
    if request.method == "POST":
        productTitle = request.form['title']
        productDescription = request.form['description']
        query = PreProcessing(productTitle)
        query = [query]
        model1, model2, model3 = Predict_Query_SVM(query)
        unique_Labels_Ctg1, unique_Labels_Ctg2, unique_Labels_Ctg3 = Get_Unique_Labels()
        Ctg1 = decode_cat01(int(model1), unique_Labels_Ctg1)
        Ctg2 = decode_cat02(int(model2), unique_Labels_Ctg2)
        Ctg3 = decode_cat03(int(model3), unique_Labels_Ctg3)
        return render_template('Products.html', title=productTitle, description=
productDescription,
                                Category_1=Ctg1[2:-2], Category_2=Ctg2[2:-2], Category_3=Ctg3
[2:-2])

if __name__ == "__main__":
    app.run(debug=True)

```

- **Utilities (Utilities.py)**

```
import re
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

def PreProcessing(text):
    text = re.sub(r'\W', ' ', text)
    text = text.lower()
    text = re.sub(r'\s+', ' ', text)
    return text

def featureSelection(data):
    tfidf = TfidfVectorizer(max_features=5000)
    X = tfidf.fit_transform(data).toarray()
    return X

def Predict_Query_SVM(query):
    model1, model2, model3 = Load_Svm_Models()
    prediction1 = model1.predict(query)
    prediction2 = model2.predict(query)
    prediction3 = model3.predict(query)
    return prediction1, prediction2, prediction3

def Get_Unique_Labels():
    unique_Labels_Ctg1 = pd.read_csv('labels_ctg1.csv')
    unique_Labels_Ctg2 = pd.read_csv('labels_ctg2.csv')
    unique_Labels_Ctg3 = pd.read_csv('labels_ctg3.csv')
    return unique_Labels_Ctg1, unique_Labels_Ctg2, unique_Labels_Ctg3

def decode_cat01(number, unique_label_c1):
    return unique_label_c1.loc[unique_label_c1['encoded'] == number, 'category'].values

def decode_cat02(number, unique_label_c2):
    return unique_label_c2.loc[unique_label_c2['encoded'] == number, 'category'].values

def decode_cat03(number, unique_label_c3):
    return unique_label_c3.loc[unique_label_c3['encoded'] == number, 'category'].values

```

- **KNN Imputation (KNNImpute.py)**

```
import pandas as pd
from sklearn.impute import KNNImputer

def impute(data):
    imputer = KNNImputer(n_neighbors=5)
    data_imputed = imputer.fit_transform(data)
    return pd.DataFrame(data_imputed, columns=data.columns)

```

B Data Samples

- **Sample Training Data (data_train.csv)**

title	description	category
"Wireless Mouse"	"A high precision wireless mouse"	"Electronics"
"Running Shoes"	"Comfortable running shoes for all terrains"	"Footwear"
"Organic Shampoo"	"Natural and organic shampoo for all hair types"	"Personal Care"

- **Sample Validation Data (data_valid.csv)**

title	description	category
"Bluetooth Speaker"	"Portable Bluetooth speaker with high-quality sound"	"Electronics"
"Hiking Boots"	"Durable hiking boots for outdoor adventures"	"Footwear"
"Herbal Conditioner"	"Herbal conditioner for smooth and shiny hair"	"Personal Care"

C Model Training and Evaluation

- **Support Vector Machine Training (Support Vector Machine.ipynb)**

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import pandas as pd

data = pd.read_csv('data_train.csv')
X = featureSelection(data['title'] + ' ' + data['description'])
y = data['category']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = SVC(kernel='linear')
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

```

- **Decision Tree Classifier Training (Decision Tree Classifier.ipynb)**

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import pandas as pd

data = pd.read_csv('data_train.csv')
X = featureSelection(data['title'] + ' ' + data['description'])
y = data['category']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

```

- **Random Forest Classifier Training (Random Forest Classifier.ipynb)**

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import pandas as pd

```

```
    data = pd.read_csv('data_train.csv')
    X = featureSelection(data['title'] + ' ' + data['description'])
    y = data['category']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    model = RandomForestClassifier(n_estimators=100)
    model.fit(X_train, y_train)

    y_pred = model.predict(X_test)
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("Classification Report:\n", classification_report(y_test, y_pred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

```

D Web Interface Templates

- **Home Page Template (HomePage.html)**

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Product Title Classification</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
</head>
<body>
    <nav class="navbar navbar-default navbar-expand-lg fixed-top custom-navbar">
        <button class="navbar-toggler" type="button">
            <!-- Navbar content -->
        </button>
        <div class="navbar-collapse">
            <ul class="navbar-nav">
                <li class="nav-item nav-custom-link">
                    <a class="nav-link" href="/upload-product">Upload Product</a>
                </li>
                <li class="nav-item nav-custom-link">
                    <a class="nav-link" href="/analysis">Model Analysis</a>
                </li>
            </ul>
        </div>
    </nav>
    <section id="hero">
        <div class="container">
            <div class="row">
                <div class="col-md-5 content-box hero-content">
                    <span>Product Title Classification</span>
                    <h1>Revolutionize Your Product Management with Smart Categorization</h1>
                    <p>We categorize your product based on Title and Description</p>
                    <a href="#" class="btn btn-regular">Learn more</a>
                </div>
                <div class="homepage col-lg-7">
                    <img src="{{ url_for('static', filename='images/main-banner.png') }}" class=
    "img-fluid" alt="classification">
                </div>
            </div>
        </div>
    </section>
</body>
</html>

```

- **Upload Product Page Template (UploadProduct.html)**

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Upload Product</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
</head>
<body>
    <div id="Main">
        <h1>Upload Product</h1>
        <form action="/product" method="POST">
            <div class="mb-3">
```

```html
                <label for="formFile" class="form-label">Title</label>
                <input class="form-control" name="title" type="text" id="formFile" required placeholder="Enter Product Title"/>
            </div>
            <div class="mb-3">
                <label for="formFileMultiple" class="form-label">Description</label>
                <textarea class="form-control" name="description" rows="10" cols="50" id="formFileMultiple" multiple placeholder="Enter Product Description for more accurate results"></textarea>
            </div>
            <button type="submit" class="btn btn-info">Submit</button>
        </form>
    </div>
</body>
</html>

```

- **Product Results Page Template (Products.html)**

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Product Classification Results</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
</head>
<body>
    <div id="Main">
        <h1>Product Classification Results</h1>
        <p><strong>Title:</strong> {{ title }}</p>
        <p><strong>Description:</strong> {{ description }}</p>
        <p><strong>Category 1:</strong> {{ Category_1 }}</p>
        <p><strong>Category 2:</strong> {{ Category_2 }}</p>
        <p><strong>Category 3:</strong> {{ Category_3 }}</p>
    </div>
</body>
</html>

```

E Environment Setup

- **Virtual Environment Setup**

```bash
# Create a virtual environment
python3 -m venv venv

# Activate the virtual environment
source venv/bin/activate

# Install required packages
pip install flask scikit-learn pandas numpy nltk matplotlib seaborn

```

- **Project Directory Structure**

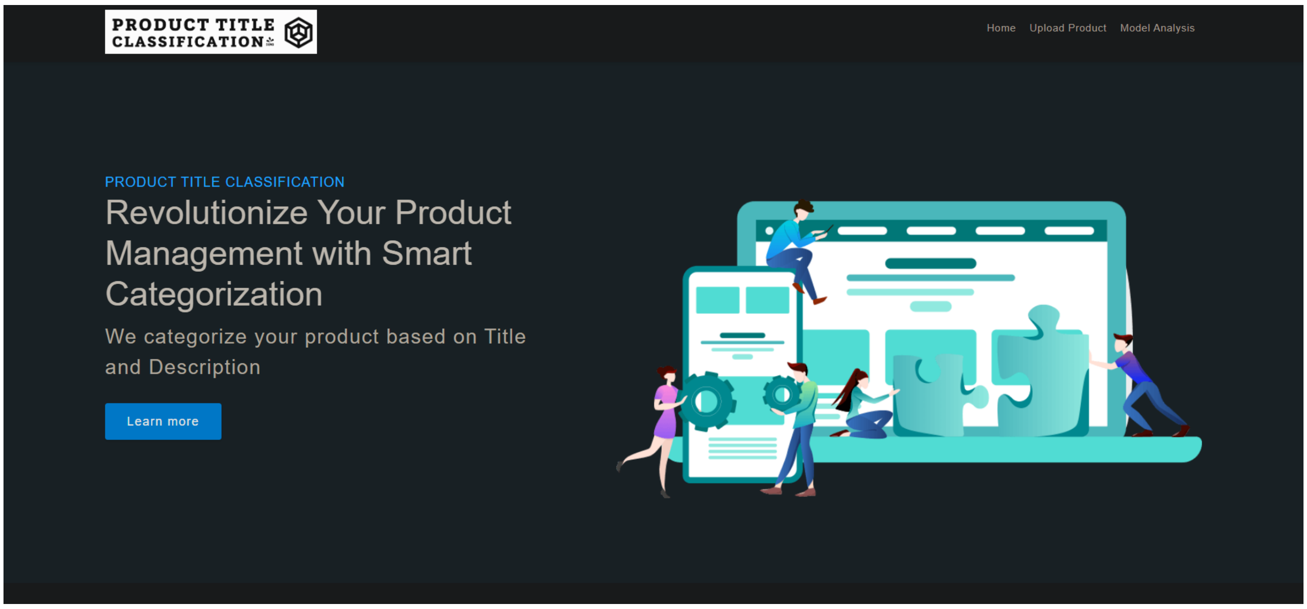
```
  Product Title Classification Source Code/
├── app.py
├── Utilities.py
├── KNNImpute.py
├── templates/
│   ├── HomePage.html
│   ├── UploadProduct.html
│   └── Products.html
├── static/
│   ├── css/
│   │   └── style.css
│   └── images/
│       └── main-banner.png
├── data_train.csv
├── data_valid.csv
└── labels_ctg1.csv
```

```
│
├── labels_ctg2.csv
├── labels_ctg3.csv
├── Support Vector Machine.ipynb
├── Decision Tree Classifier.ipynb
├── Random Forest Classifier.ipynb
└── README.md
```

F Application Screenshots

The following screenshots illustrate the functionality and user interface of the Product Title Classification web application:

- **Home Page**: The home page provides an overview of the application.



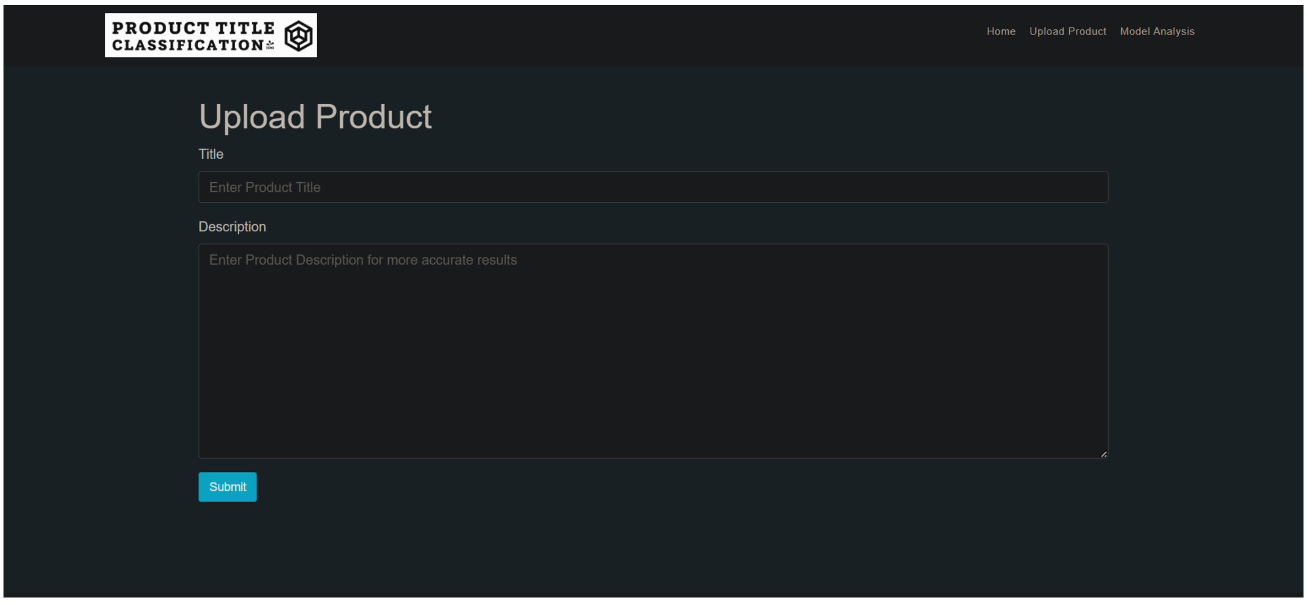
PRODUCT TITLE CLASSIFICATION

Revolutionize Your Product Management with Smart Categorization

We categorize your product based on Title and Description

Figure 3: Home Page

- **Upload Product Page**: The upload product page allows users to input product details.



Upload Product

Title

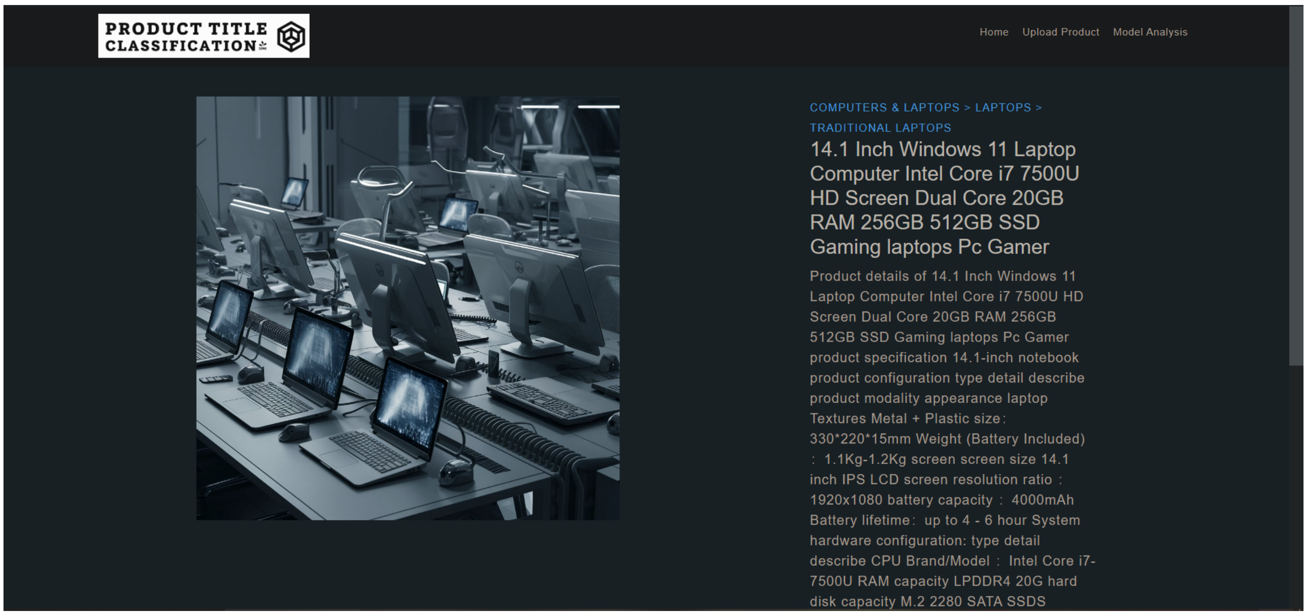
Enter Product Title

Description

Enter Product Description for more accurate results

Figure 4: Upload Product Page

- **Product Results Page**: The product results page displays the classification results.



COMPUTERS & LAPTOPS > LAPTOPS > TRADITIONAL LAPTOPS

14.1 Inch Windows 11 Laptop Computer Intel Core i7 7500U HD Screen Dual Core 20GB RAM 256GB 512GB SSD Gaming laptops Pc Gamer

Product details of 14.1 Inch Windows 11 Laptop Computer Intel Core i7 7500U HD Screen Dual Core 20GB RAM 256GB 512GB SSD Gaming laptops Pc Gamer product specification 14.1-inch notebook product configuration type detail describe product modality appearance laptop Textures Metal + Plastic size : 330*220*15mm Weight (Battery Included) : 1.1Kg-1.2Kg screen screen size 14.1 inch IPS LCD screen resolution ratio : 1920x1080 battery capacity : 4000mAh Battery lifetime : up to 4 - 6 hour System hardware configuration: type detail describe CPU Brand/Model : Intel Core i7-7500U RAM capacity LPDDR4 20G hard disk capacity M.2 2280 SATA SSDS

Figure 5: Product Results Page