Consumer Experience Project

Detailed Project Report

Written By	Shraddha Sawant
Document Version	1.0
Last Revised Date	24-Nov-2023

Table of Contents

Abstract

The objective of this project is to identify the experience of consumers. In the market where there are number of choices to select from for each product and service, it becomes important for every company to consistently keep on providing good service to its consumers. Each company gets large amount of feedback reviews from its customers everyday. It is impossible to manually go through each review and find out if the feedback is positive or not. So this is the problem that major companies face today. Also, manually identify positive and negative reviews is not possible.

Hence, the aim of this project is to automatically predict if the customers reviews are positive or negative. This knowledge will help company identify the negative reviews, fix the problems, and increase and maintain their customer base.

Project Technical Description

The Project Directory consists of the following Modules and Classes:

1. Components
2. Entity
3. Pipeline
4. Config File
5. Logger File
6. Exception File
7. Uitls File
8. Predictor File

Let us discuss each component in detail.

Components

Components Module consists of 5 Classes as below:

- data_ingestion
- data_transformation
- model_trainer
- model_pusher
- model_evaluation

1. **data_ingestion**:

Data _ingestion class is created to intake data from database and convert it into Train and Test Files respectively. This class consists of two methods, the first method called as the __init__ method is run as soon as the object of class is instianted. The __init__ attributes are shared by all the instances of class data_ingestion. In this class we have not used the concept

Of inheritance or polymorphism. But if needed, we can create the same.

The second method of class data ingestion is called as the initiate_data_ingestion. We perform following set of instructions in this class.

We first call the helper function utils, from this the method get_collection_as_dataframe. This method, when passed the attributes to collection name and database name connects to mongo client and takes dataframe as input in a temporary variable df. We can also directly chose to connect to csv file if needed. We will discuss the properties of utils function while discussing utils function in depth. Once the data is inside function, we store it in variable df whose type is of pandas dataframe. Once, we get our data in dataframe format we then remove duplicates, remove null values. This makes our dataset free of any redundant information. Then we call the method train_test_split from class sklearn.prepocessing. Here we, split the dataframe into two files Train File and Test File respectively. We then convert the dataframes into CSV files and store them data_ingestion_artifact paths respectively.

2**. data_transformation**:

The second main class in our project is the data transformation class. This class also consists of two methods. The first method __init__ is for initializing the attributes throughout the class. The second method is called as the initiate_data_transformation. Once this method is called, it works as follows.We first input the Train and Test Files from the data_ingestion_artifact file paths and store them into temporary variables. Then we split Train Files and Test Files into Input feature and Target Features so that we can apply data transformations on them respectively. We do this by using the slicing feature from pandas. We then apply Label Encoder on Target Feature and CountVectorizer on Input Feature. Before applying CountVectorizer on Input Feature we clean it using Regular Expression and Custom Function to remove stop words, punctuations, Full stops, Exclamations, tags etc. Once feature engineering is done on input set and encoding is performed on target set, we concatenate them using numpy concat method and save them into temporary variables. Now, we what we have is Train array and Test array. We save the

Train array and test array in train npz and test npz format respectively and send them to data_transformation_artifact file paths respectively.

3. **model_trainer**:

The model_trainer class, as the name suggests trains the model. This class consists of three methods. The first is the constructor method, the second is train_model method and the third method is the initiate_model_trainer_method. In the constructor method, we are initializing the data_transformation_artifact, model_trainer_config. We will use the attributes called and instantiated in the method throughout this class. Next we create method train_model. We have used RandomForestClassifier for this project. From the analysis we carried out in out Jupyter Notebook, we received the highest accuracy for this classifier model. In the third method, we initiate model training. The type of our model_training function is of model_trainer_artifact type.

Firstly, we store the output of the data_transformation_artifact file in temporary variables. The input that we have is in array format. Because, the data needs to be in integer format when training. We create an object of RandomForestClassifier and fit it on our train_input and train_target_feature.

We then call the utils save_object to save it in model_trainer_config_path and Then lastly we give this model object to be saved in model_trainer_artifact path.

5. model_pusher:

The we have the mod_pusher class. This class consists of two methods, constructor method and model_pusher method. In this class, we are saving the models into two file paths. First file path is the model_pusher_config_model_path and the second file_path is the model_pusher_saved_model_dir file path.

5**. model_evaluation**:

The last class of the components directory is the model_evaluation class. In this class, we are calling models from two file paths one is model_trainer_config path and second is saved_model_directory file path.Because, when we create the File Training_Pipeline, we are taking input from the user, transforming it, training it and saving it into model_pusher_config file path. In Model Evaluation, we are creating the objects for the two model paths, previously trained model and currently trained model. We will check the accuracy of two models and whichever model accuracy is higher we have to check why is it acceptable according to business requirement. If it is acceptable then we have to continue using the model if not we have to do Jupyter notebook analysis and find out the reasons accordingly for future use.

Entity

Components Module consists of 2 Classes as below:

1. artifact_entity
2. config_entity

1. artifact_entity:

This class is made to store all the variables which are used as output attributes obtained in all the individual classes from components module.

2. config_entity:

This class is made up of all the directory paths which are used as inputs in all the classes in components module.

Pipeline

Pipeline Module consists of 1 Class as below:

1. training_pipeline

1. training_pipeline:

This class is made to call all the classes in such a way that the end result is the prediction output for the previously trained class and currently trained class.

Helper Files

Rest all the files are the helper files which are called at various times and instances while working with all the classes. The helper files are as below:

1. Logger File
2. Exception File
3. Predictor file
4. Config File
5. Utility File

1. logger.py:

This class helps us to create logs in our project directory. It creates logs according to the date and timestamp.

2. exception.py:

This class helps us to raise exception according to INFO Level.

3. config.py:

This class calls pymongo client module and stores database url for use throughout the project.

4. utils.py

This class stores various functions to create, save and load objects.

5. predictor.py

This class contains Model Resolver class to store all the file paths.

Data Analysis and Model Building

The first step in creating our project is doing data analysis and then model building. Data Science Projects consists of 3 phases, Data Preprocessing, Feature Extraction/Data Vectorization and Model Building.

In our Project, we have applied Text Preprocessing, Text Vectorization and then Model Building. Let us see all the steps in detail:

1. **Text Preprocessing**
2. **Text Vectorization**
3. **Model Building**

1. **Text Preprocessing:**

Text Preprocessing is done to remove conversational style from the string. Our dataset is of reviews. Review writing is an informal style and hence

may contain certain quotation marks, full stops, commas, punctuations, double quotes, urls, capital letters etc. It may also contain stop words. It may also contain html tags, brackets etc. It may also contain special characters. Our machine learning algorithms cannot work with such type of data. Hence, we have to remove all the special characters from our data before we feed it to a machine learning algorithm.

The libraries we have used are pandas, regular expression and ntlk.

2. Text Vectorization

Once all the special characters are removed and all the words are in one case such as either lowercase or uppercase, we can apply different text vectorization techniques to convert them into integer format. This method is called as text vectorization. It is done by applying OneHot Encoding, BagOfWords, Ngrams or TF-IDF. In this project, BagOfWords, Ngrams and TF-IDF is used and is then is checked against each of the Classifier algorithms for its accuracy.

2. Model Building

Once text data is converted into Numeric data, classifier models including Naïve Bayes, Decision Trees, Random Forest Classifier and Support Vector Classifier is applied on each of the text vectorizor technique and is checked for its accuracy.

At the end, we found out that Random Forest Classifier with Count Vectorizer gives us the best result. Hence, we used it in our final project Implementation.

END