

Settyl Data Science And Machine Learning Engineer Task

I have used the jupyter notebook for Data Preprocessing, Model Development, Model Training and Evaluation here the steps involved starting with .

1) Understanding the Data :

the dataset has two columns internal status target column and external status independent column with 1222 observations and both the columns was object type.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1222 entries, 0 to 1221
Data columns (total 2 columns):
#   Column          Non-Null Count  Dtype
---  -
0   externalStatus  1222 non-null   object
1   internalStatus  1222 non-null   object
dtypes: object(2)
memory usage: 19.2+ KB
```

Fig 1: Dataset information

2) Preprocessing the data:

- As observed there were no missing values.
 - Next step was to cleaning the external status column
 - Removed unwanted characters
 - Converted text to lowercase
 - Removed unwanted spaces
 - Removed stopwords
- Encoded the internal status target column using label encoder and also visualize the multiclass label in the target column and their frequency using barplot.

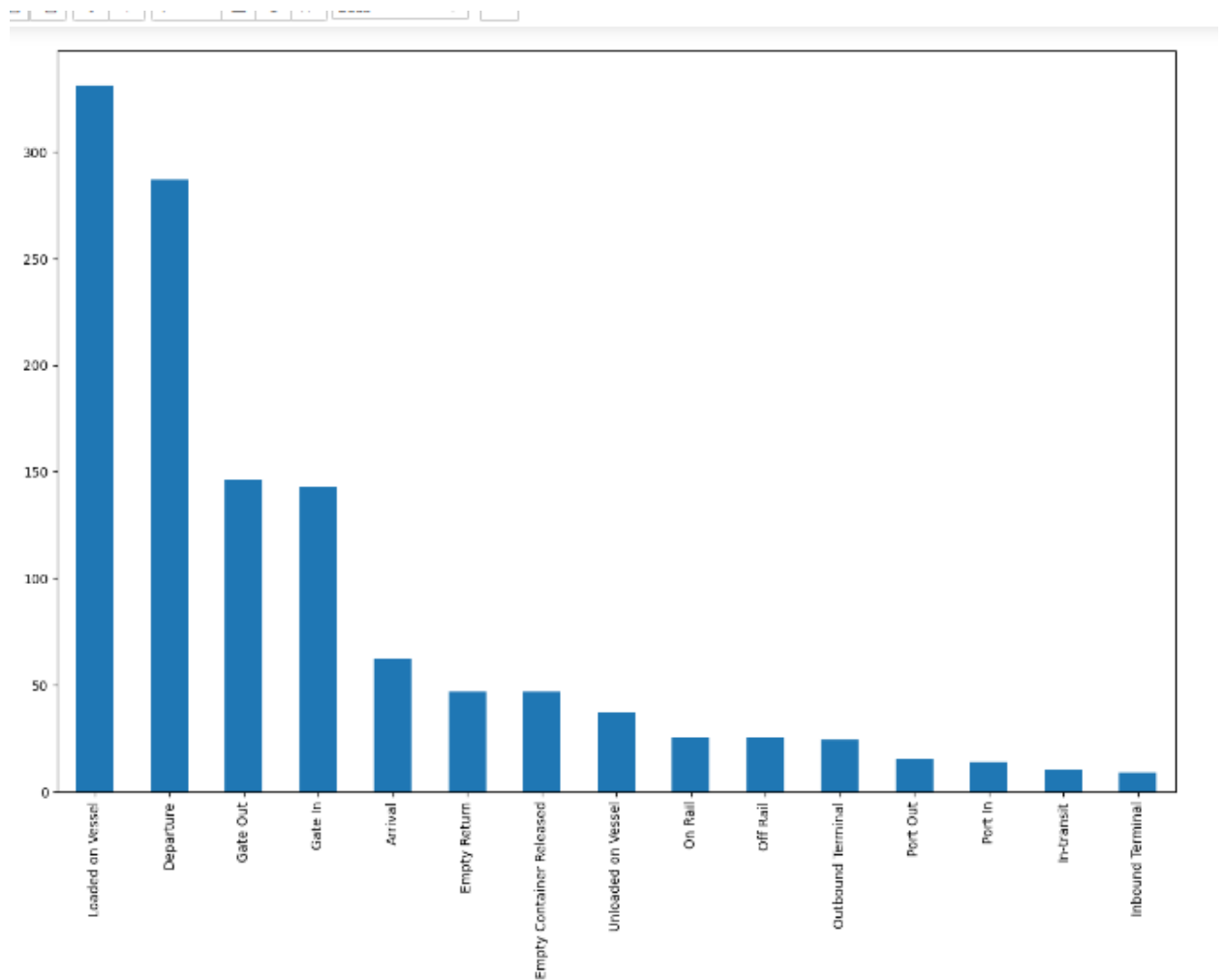


Fig 2: target column class label Frequency

- 'Loaded on vessel' class label has the highest frequency and 'inbound terminal' has the least frequency.

3) Model Development:

1.Data Splitting:The dataset is divided into training and testing sets using the `train_test_split` function from scikit-learn.

2. Vectorization: The text data is converted into numerical representations using `CountVectorizer`.

3. SparseTensor Conversion: Since the vectorized data is sparse, it's converted into `SparseTensor` objects, which is a data structure TensorFlow can efficiently handle. This conversion facilitates the input of sparse data into the neural network model.

4. Model Architecture: The neural network model architecture is defined using TensorFlow/Keras. It consists of sequential layers: two dense layers with ReLU activation functions and dropout layers in between. The first layer has 128 neurons, the second layer has 64 neurons, and the output layer has neurons equal to the number of unique labels in the dataset with softmax activation for multi-class classification.

5. Model Compilation: The model is compiled with an optimizer (Adam), which is a popular choice for gradient-based optimization algorithms, a loss function (sparse categorical cross-entropy), and evaluation metrics (accuracy).

6. Model Training: The model is trained on the training data using the `fit` method. During training, the model adjusts its parameters to minimize the loss function. Validation data is also provided to monitor the model's performance on unseen data during training.

While training the model using validation data I got a accuracy of approximately 92.22% of the training samples and the model correctly predicted the labels for approximately 93.47% of the validation samples.

7. Model Evaluation: After all these training, the model is evaluated on the testing data using the `evaluate` method.

During evaluation I observed that the model achieved a test accuracy of approximately 93.47%, with a test loss of around 0.2497. This indicates that the model's performance on unseen test data is consistent with the performance observed during training and validation, suggesting that the model has successfully generalized to unseen data.

For further deployment of the trained model saved the code .

4)API Development:

For deployment created the file called app.py where it sets up a FastAPI application to serve predictions from the trained model based on external status descriptions. When the server is started, it listens for requests and responds with predictions accordingly.

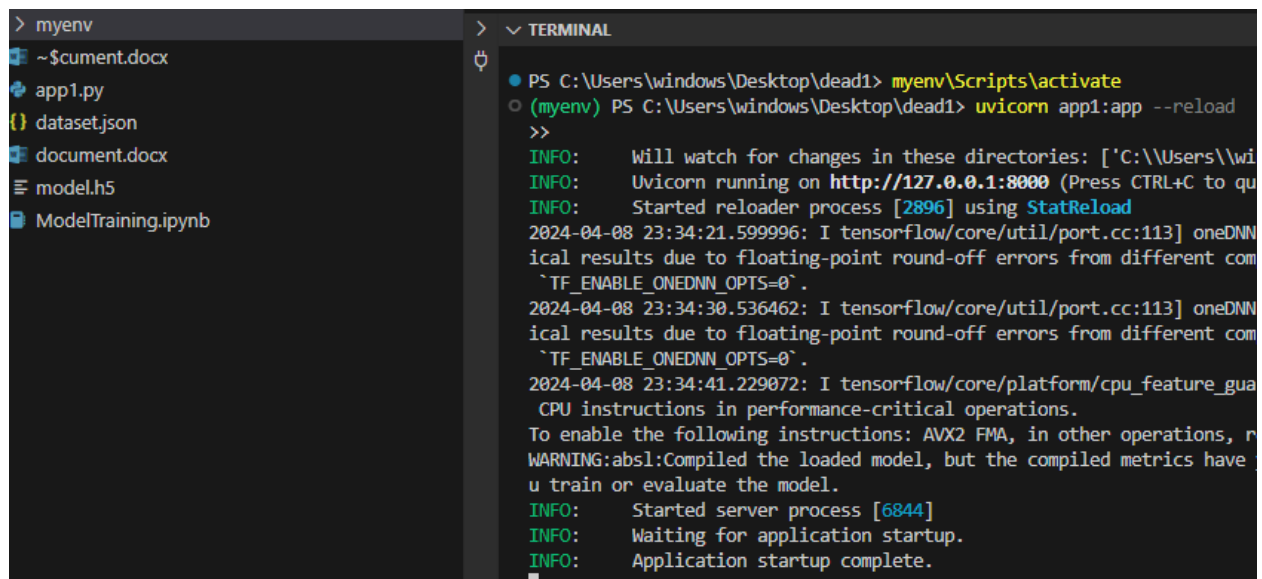
As Follows Starting with:

- Defined FastAPI app
- Loaded trained model
- /predict: This POST endpoint accepts input data then Defined prediction endpoint
- Finally started the FastAPI server using uvicorn.
- As uvicorn helps us to run Asynchronous Server Gateway Interface applications.

At last created and activated the my virtual environment and installed all the required libraries and packages, further more passed the command to run the FastAPI application named app1 located in the file app1.py.

And got the URL '**http://127.0.0.1:8000**'.

By using this above URL we can access the application in web browser or make requests to it.



```

> myenv
~$cument.docx
app1.py
dataset.json
document.docx
model.h5
ModelTraining.ipynb

> TERMINAL
PS C:\Users\windows\Desktop\dead1> myenv\Scripts\activate
(myenv) PS C:\Users\windows\Desktop\dead1> uvicorn app1:app --reload
>>
INFO: Will watch for changes in these directories: ['C:\\Users\\wi
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to qu
INFO: Started reloader process [2896] using StatReload
2024-04-08 23:34:21.599996: I tensorflow/core/util/port.cc:113] oneDNN
ical results due to floating-point round-off errors from different com
`TF_ENABLE_ONEDNN_OPTS=0`.
2024-04-08 23:34:30.536462: I tensorflow/core/util/port.cc:113] oneDNN
ical results due to floating-point round-off errors from different com
`TF_ENABLE_ONEDNN_OPTS=0`.
2024-04-08 23:34:41.229072: I tensorflow/core/platform/cpu_feature_gua
CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, r
WARNING:absl:Compiled the loaded model, but the compiled metrics have
u train or evaluate the model.
INFO: Started server process [6844]
INFO: Waiting for application startup.
INFO: Application startup complete.

```

Fig 3: Activating the virtual environment

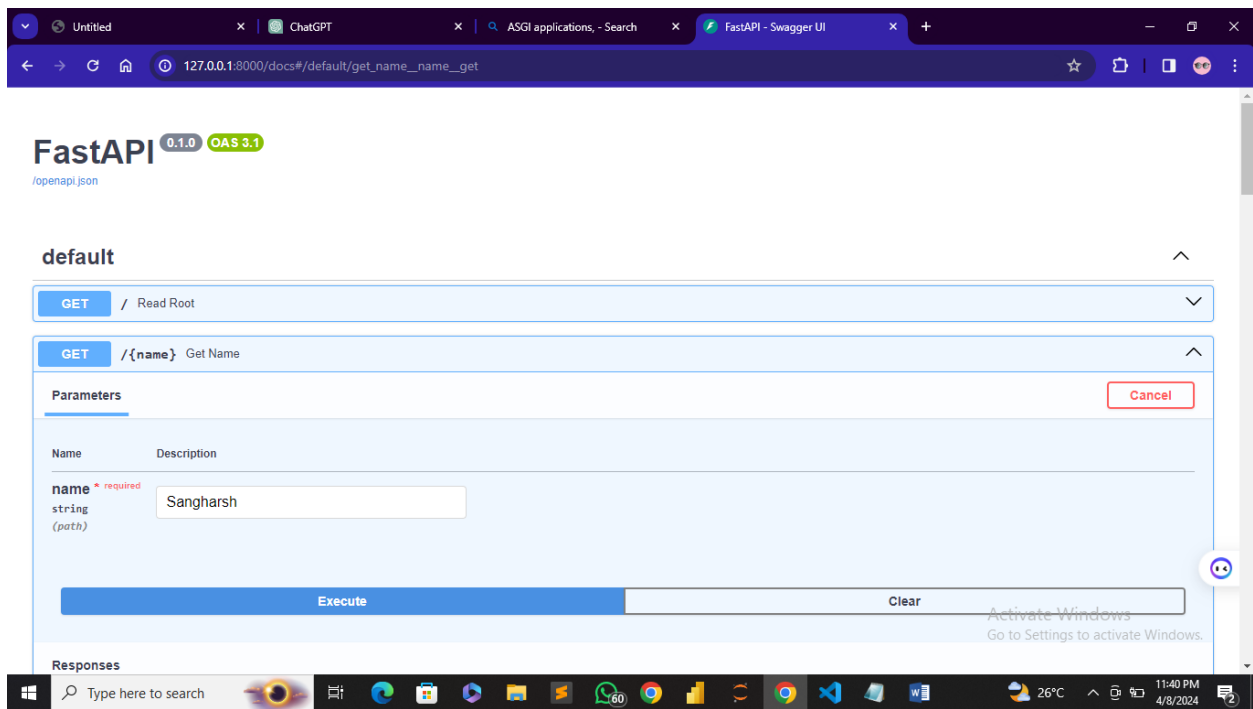


Fig 4: Snapshot of the app

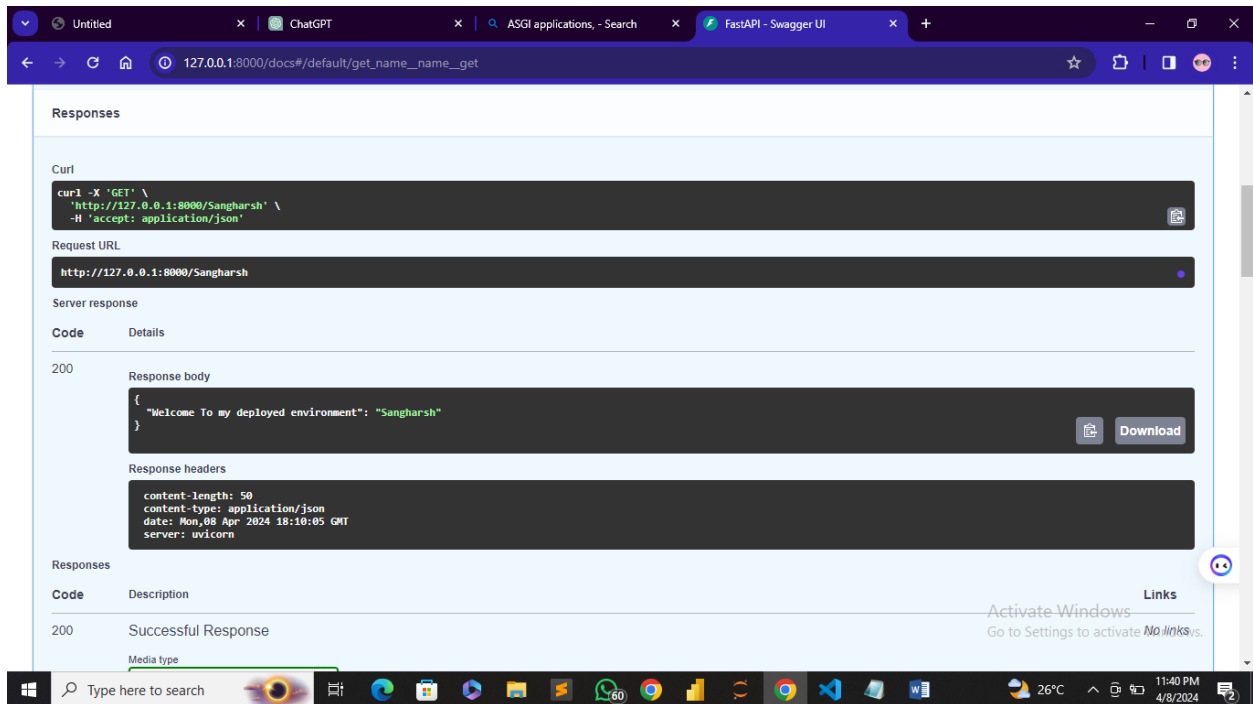


Fig 5: Giving the user name

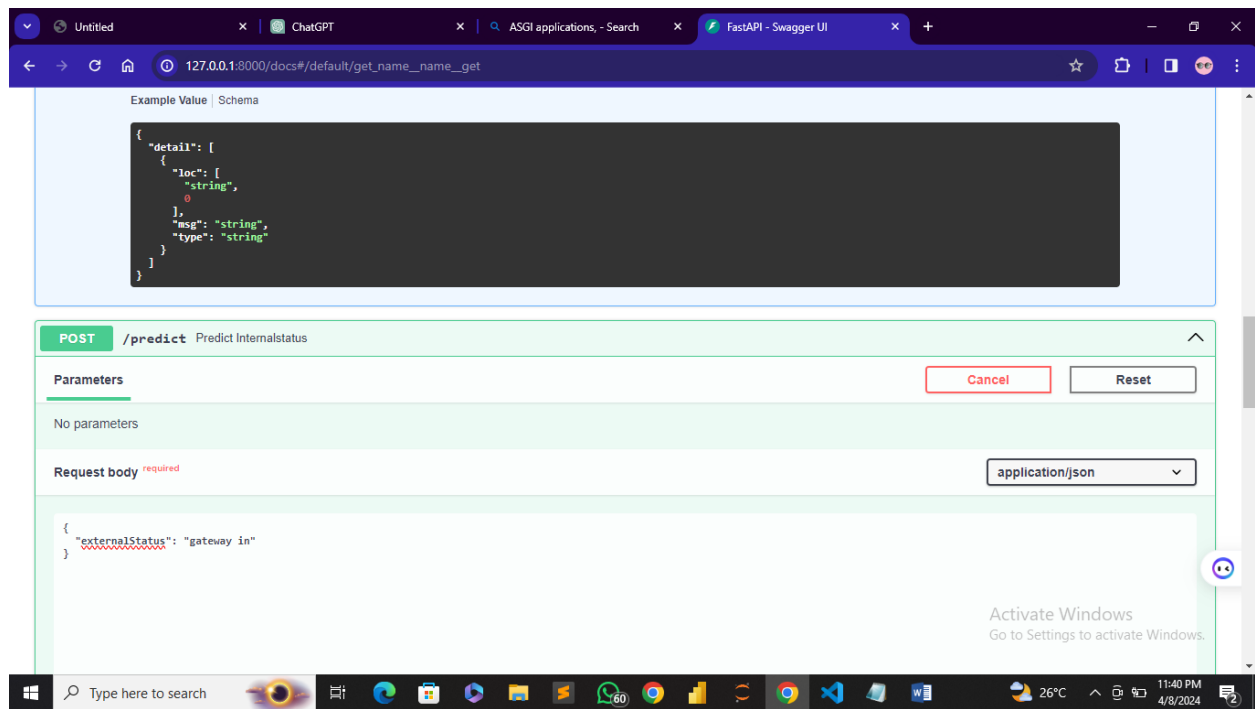


Fig 6: Passing the input text external status