
# Java project integration with ML model for prediction

By now, we have already completed the ML project and created a ML model. In this document, we are sharing the process of integrating that ML model from JAVA application for prediction.

For doing this, we need the knowledge of **Flask**, apart from Python and Java.

# Flask Integration -

**Flask is a web application framework** written in python. We need to use Flask api to get the prediction from our machine learning pre-trained model.

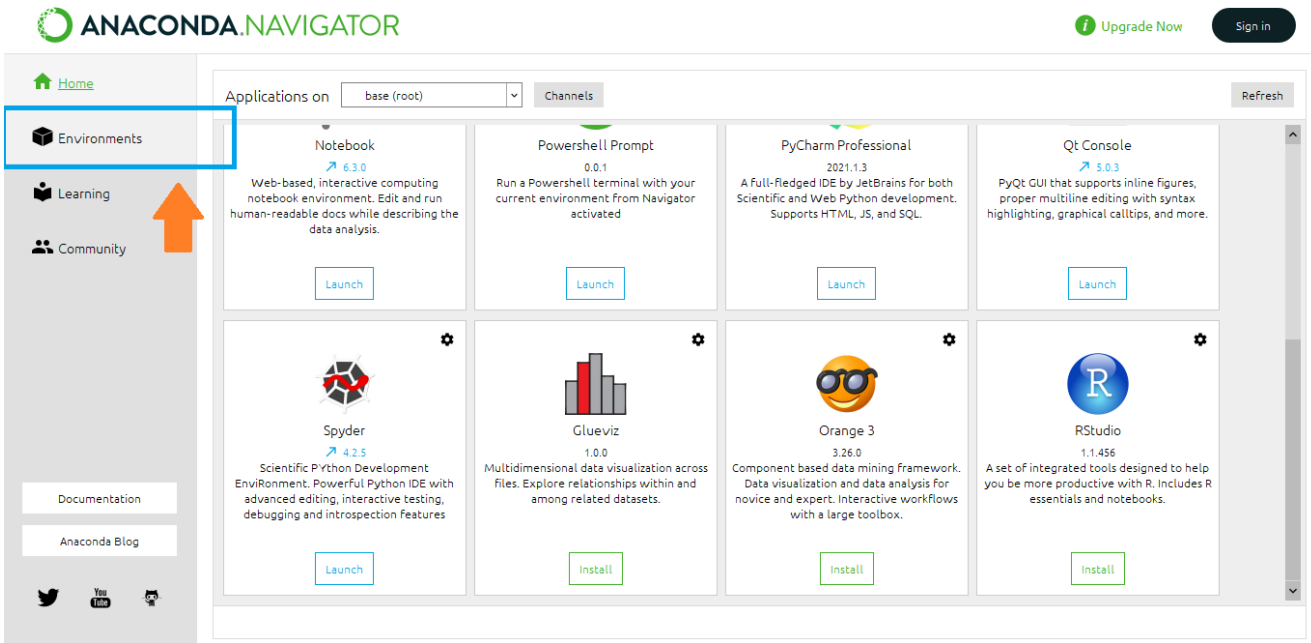
## Note : Use your model file which you have created in ML Project. Keep the name of the model file as “model.sav”

```
model = pickle.load(open("model.sav", 'rb'))
final_result = model.predict(nulldata2)
final_result = pd.Series(final_result,name='avg_delay')
```

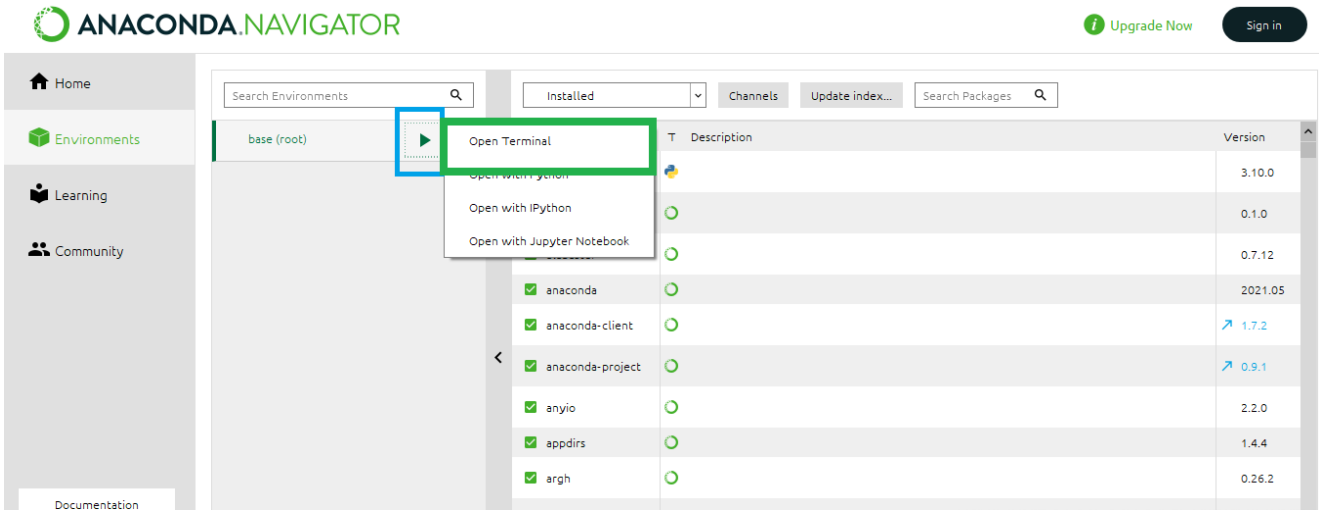
## Steps 1 - Check for the Flask package

Flask Package generally comes with the anaconda by default. To check if flask present or not

we can open terminal of anaconda -

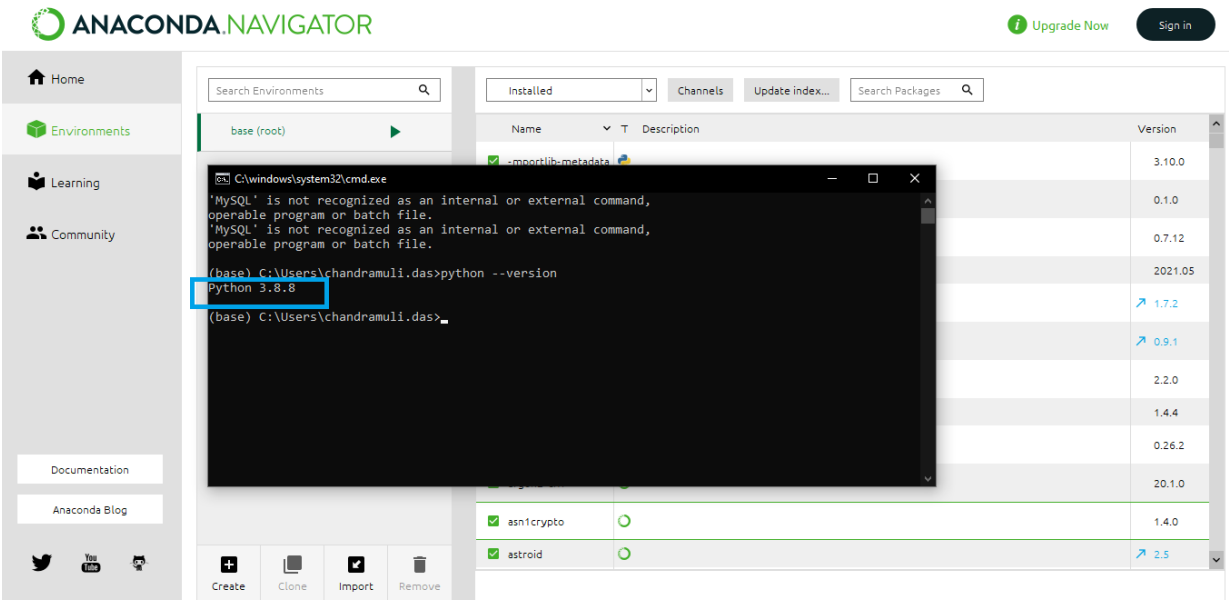



Click on the **environment option**, at the left side, then click on the play button right beside the base environment, and need to open the terminal.



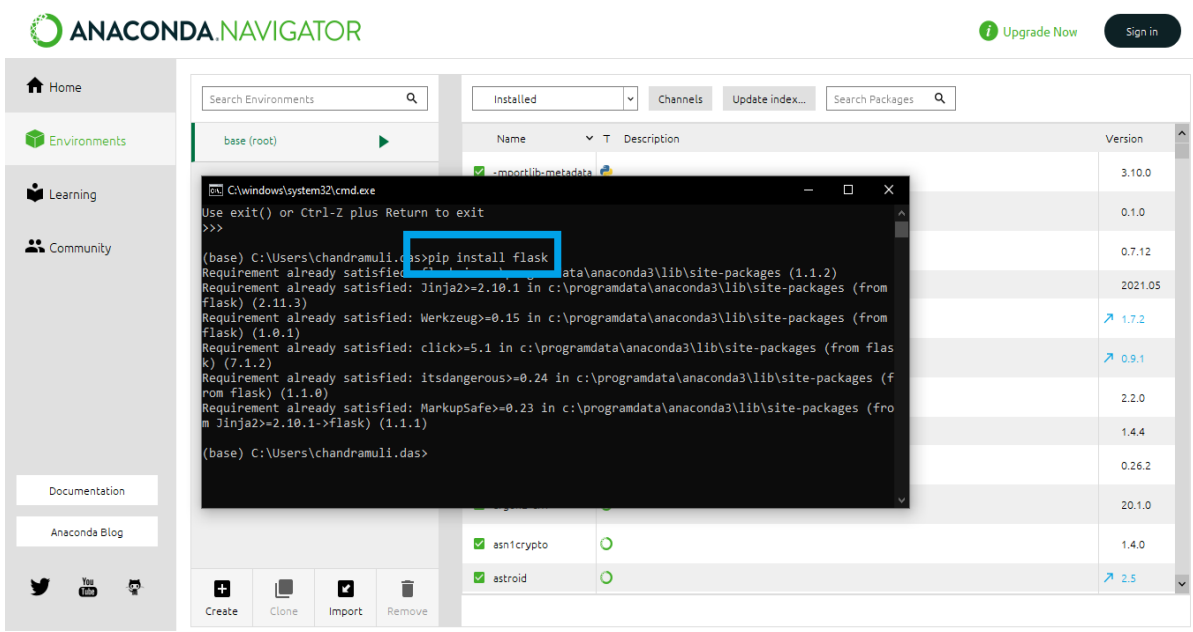

3. Open the terminal, and check for the python version. The python version should be **Python 3.8.**

```
>python --version
```



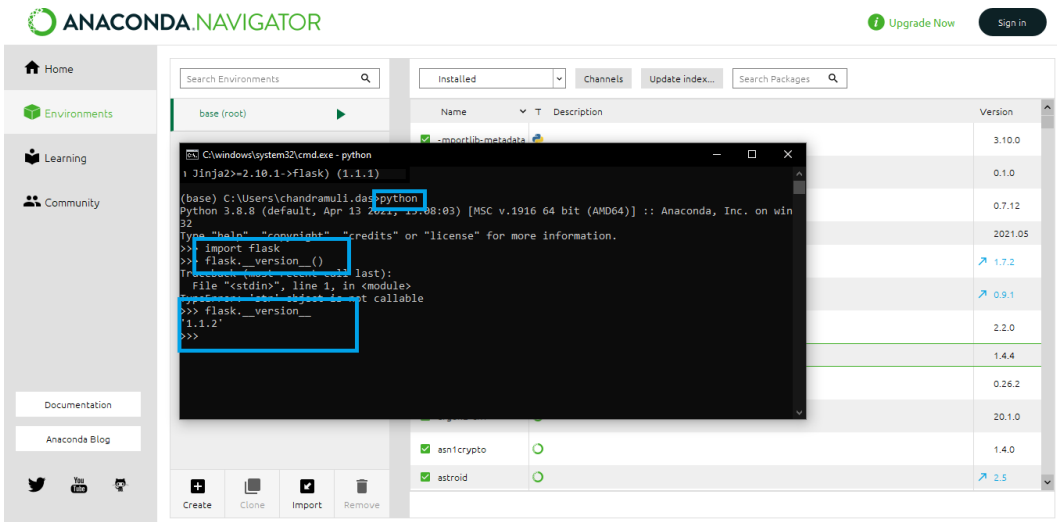

4. Check for the Flask -

Command - **pip install flask**



a If the flask is not pre installed, it will start the installation

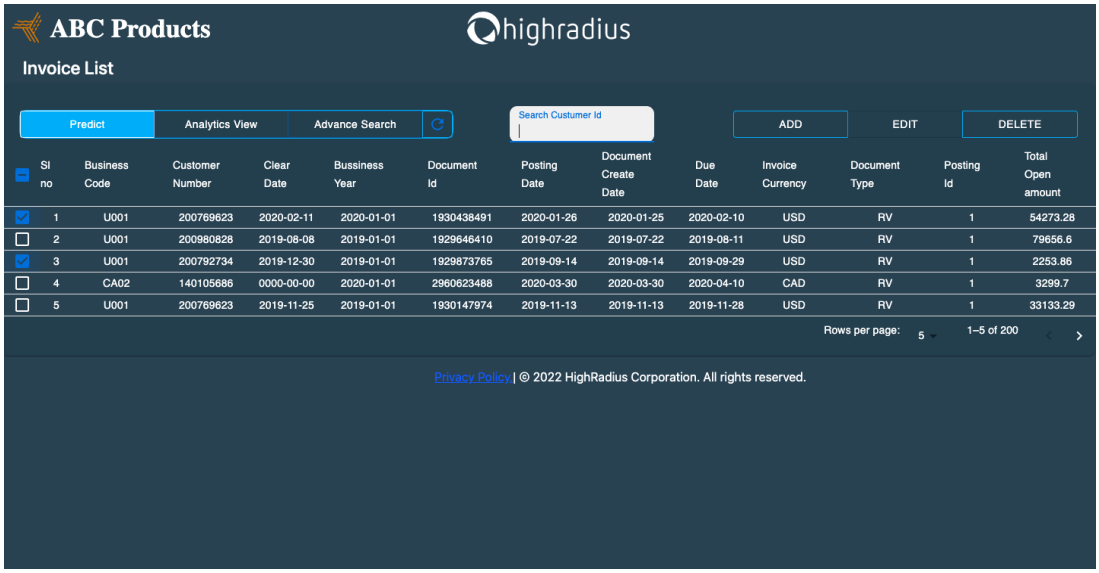
b. If the flask is already installed, it will return **"Requirement Already Satisfied"**



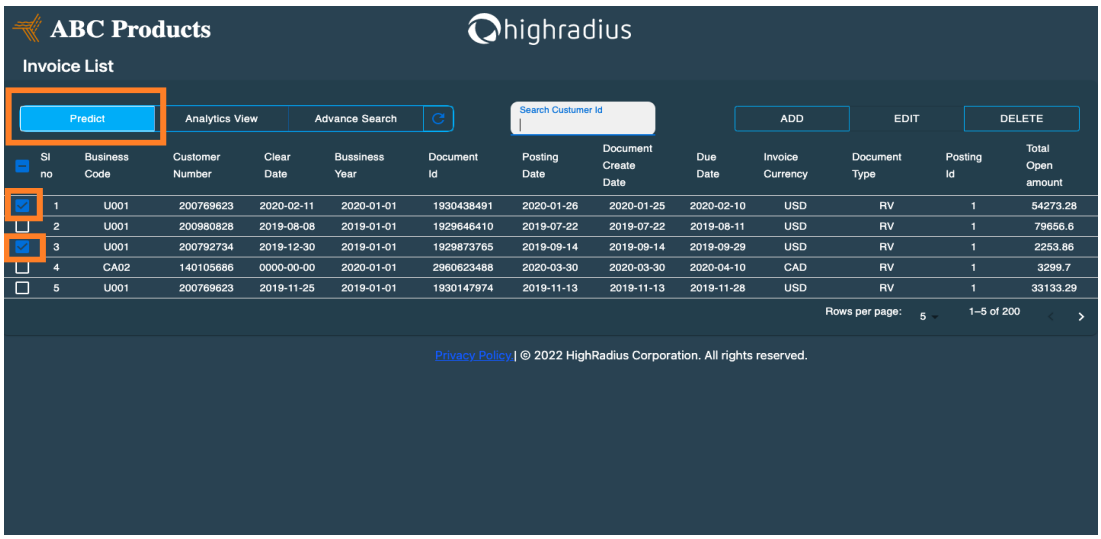

c. The flask version should be **1.1.2**

## Step 2- Prediction -

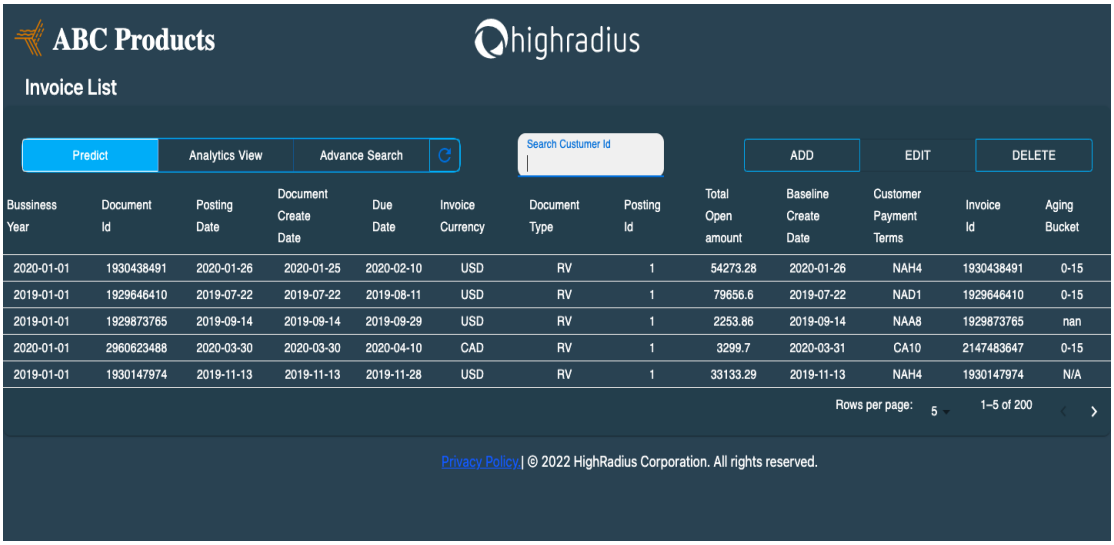
The default page is following -




Select the number of rows for which we need to predict the Aging Bucket. Click on the **Predict** button.




Once we click the **Predict** button, it will populate the prediction on the desired row as shown below:




| Bussiness Year | Document Id | Posting Date | Document Create Date | Due Date | Invoice Currency | Document Type | Posting Id | Total Open amount | Baseline Create Date | Customer Payment Terms | Invoice Id | Aging Bucket |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2020-01-01 | 1930438491 | 2020-01-26 | 2020-01-25 | 2020-02-10 | USD | RV | 1 | 54273.28 | 2020-01-26 | NAH4 | 1930438491 | 0-15 |
| 2019-01-01 | 1929646410 | 2019-07-22 | 2019-07-22 | 2019-08-11 | USD | RV | 1 | 79656.6 | 2019-07-22 | NAD1 | 1929646410 | 0-15 |
| 2019-01-01 | 1929873765 | 2019-09-14 | 2019-09-14 | 2019-09-29 | USD | RV | 1 | 2253.86 | 2019-09-14 | NAA8 | 1929873765 | nan |
| 2020-01-01 | 2960623488 | 2020-03-30 | 2020-03-30 | 2020-04-10 | CAD | RV | 1 | 3299.7 | 2020-03-31 | CA10 | 2147483647 | 0-15 |
| 2019-01-01 | 1930147974 | 2019-11-13 | 2019-11-13 | 2019-11-28 | USD | RV | 1 | 33133.29 | 2019-11-13 | NAH4 | 1930147974 | N/A |

## The Flask code -

Once we trigger the prediction, the following code will run, and the following code will run.

In [1]:

```
from flask import Flask, redirect, url_for, render_template, request,jsonify, make_response
import New_Bucket
import pandas as pd
data = pd.DataFrame()
app = Flask(__name__)

@app.route("/",methods=["POST", "GET"])
def home():
    if request.method == "POST":
        print(request.form)
        business_code = request.form["business_code"]
        cust_number = request.form['cust_number']
        name_customer = request.form['name_customer']
        clear_date = request.form['clear_date']
        buisness_year = int(request.form['buisness_year'])
        doc_id = int(request.form['doc_id'])
        posting_date = request.form['posting_date']
        due_in_date = request.form['due_in_date']
        baseline_create_date = request.form['baseline_create_date']
        cust_payment_terms = request.form['cust_payment_terms']
        converted_usd = float(request.form['converted_usd'])
        data['business_code'] = [business_code]
        data['cust_number'] = [cust_number]
        data['name_customer'] = [name_customer]
        data['clear_date'] = [clear_date]
        data['buisness_year'] = [buisness_year]
        data['doc_id'] = [doc_id]
        data['posting_date'] =[posting_date]
        data['due_in_date'] = [due_in_date]
```

Click here to access the **Master folder.** After a successful run the desired output will populate the column.

Steps to Run -

1. Download the folder from the above link
2. Open the integration.py file in jupyter notebook
3. From the Payment Date Prediction notebook file, save the final model.
4. Rename the **your ML Project** model name as **"model.sav"**
5. Store the **"model.sav"** into the master folder.
6. Run the Jupyter Notebook.