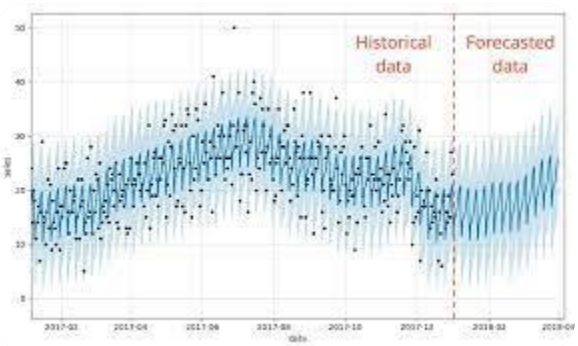


PHASE 5

ABSTRACT

Phase 5: Deployment and Productionization

Soumen Chatterjee

Phase 5 – 06-Mar-2022

- At this stage we will work on **Deployment and Productionization of the best model**

There are different approaches of deploying a model – these are: Locally using scikit learn/ AWS/Heroku/GCP/Azure etc.

For this particular project we tried

1. **Deploying locally**: This is the easiest way of testing like how an end user will see the model at his/her end.
   This way is good to test the model from end user prospective, but in live scenario we will need to send a link to platform which is always available and user can check the query at any time. In this case there is no such option available

2. **Deploying at AWS:** This approach is something actually being used in live environment. This is one of the ways used in live environment other ways are Heroku, GCP, Azure.
   As we have used the free tier resources available, so the resources available is very limited. And for a model which is heavy in nature might get timeout

   Deploying model in AWS is a bit easy compared to others (though this statement varies based on users)

To deploy the model we have used –

Flask: To create the user interface & run the model
Joblib: Used this library to zip the trained model so that we can use the same in app

Saving the Logistic Regression model as Pickle file for Production purpose

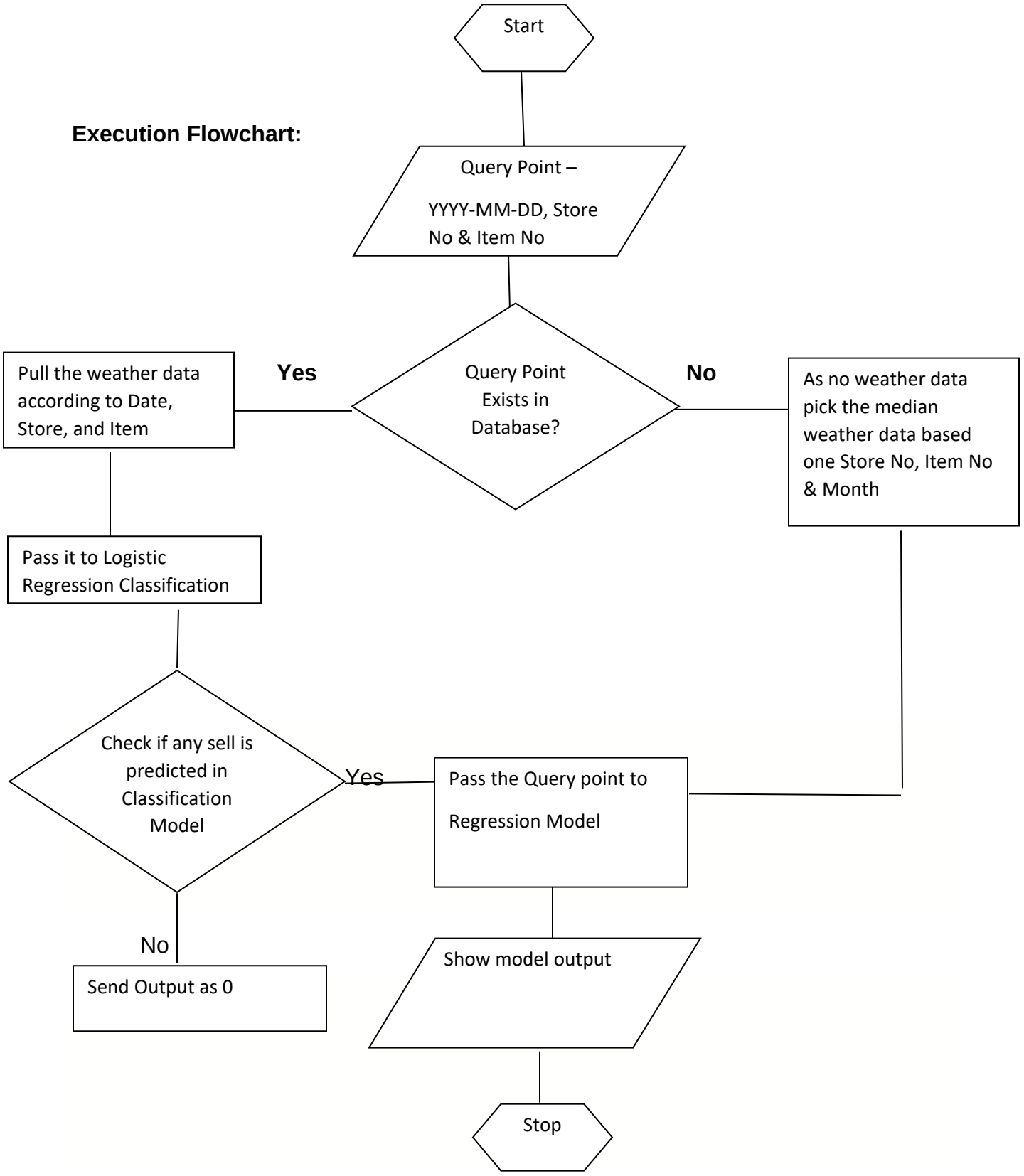
```
[ ] from joblib import dump, load
    dump(model_cls, '/content/drive/MyDrive/Project_TS_Walmart/Phase 2/Data/cls_LogisticRegrsn.pkl')

    ['/content/drive/MyDrive/Project_TS_Walmart/Phase 2/Data/cls_LogisticRegrsn.pkl']
```

Loading the .pkl file

```
# Our best model is XGBRegressor, then Random Forest.
#But for both these Model- while loading pkl file we faced error in terms of Parameter so used Linear Regression
reg_RF = joblib.load('LinearReg.pkl')
predictionReg = reg_RF.predict(QueryPoint)
FinalPrediction = predictionReg
```

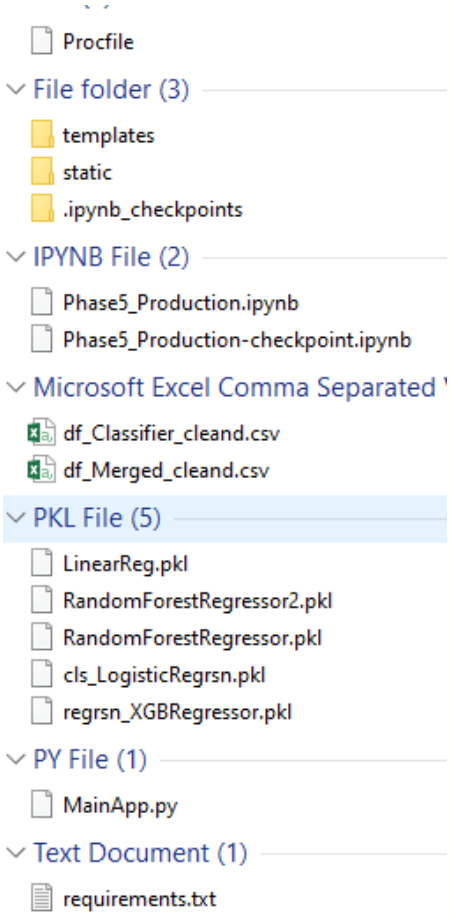
Execution Flowchart:



Below is the folder structure, which contains -

1. "**MainApp.py**" : has all the codes to read through the query point from user interface (created by Flask API)
2. "**df_Classifier_cleand.csv**", "**df_Merged_cleand.csv**": Cleaned merged data created after EDA
3. "**.pkl**" files: For all the models we created at Phase 4 modelling stage. Linear Regression classification model (LinearReg.pkl) will be used for sure, and any one of the regression models will be used.
4. "**static**" **folder**: Holds the images, static files if any
5. "**templates**" **folder**: Holds the .html files which are used for user interfaces

    a) "**querypage.html**": Is the landing page where user will input the Date, Store No, Item no as Query point
    b) "**outputpage.html**": Once the above is submitted "**outputpage.html**" will pop up with the output
    c) "**close.html**": And lastly "close.html" will be called at the end



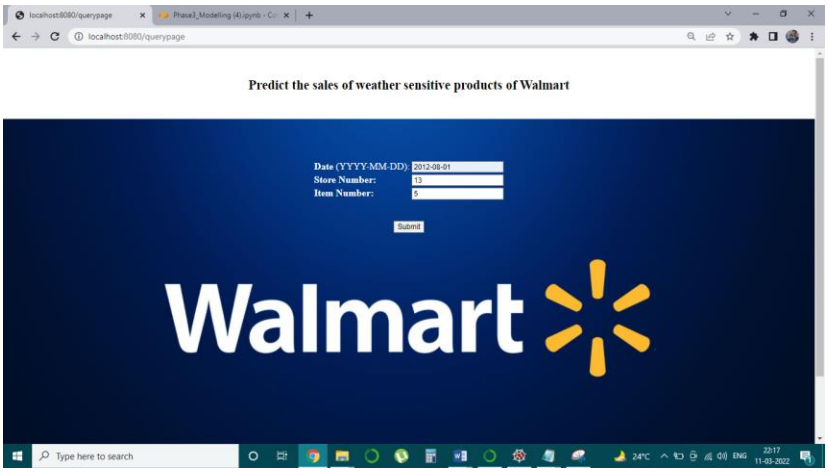
Deploying and checking the Model Locally:

1. First we will need to run the “app.py” (a code is there to load the corresponding Model’s .pkl file)
2. Then our system will act as server, based on the port no given a link will be generated. Something like below –

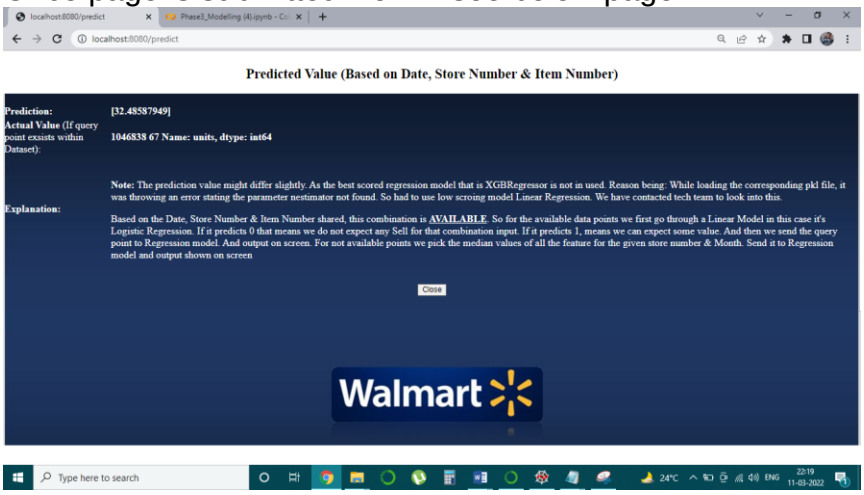
```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```

```
 * Running on http://0.0.0.0:8080/ (Press CTRL+C
to quit)
```

3. Use the link: http://localhost:8080/querypage (change the yellow part as per the port no given)
4. On first screen we will see below –



3. Need to provide the query point details date, store no, item no
4. Once page is submitted we will see below page –



Deploying the Model at AWS:

Steps:

1. Build a model on our local system and store the model and other key model related variables in .pkl files
2. Launch a micro instance on AWS.
3. Connect to the AWS box [ssh]
4. Move the files to an AWS EC2 instance/box [scp]
5. Install all packages needed on the AWS box.
6. Run app.py on the AWS box.
7. Check the output in the browser.

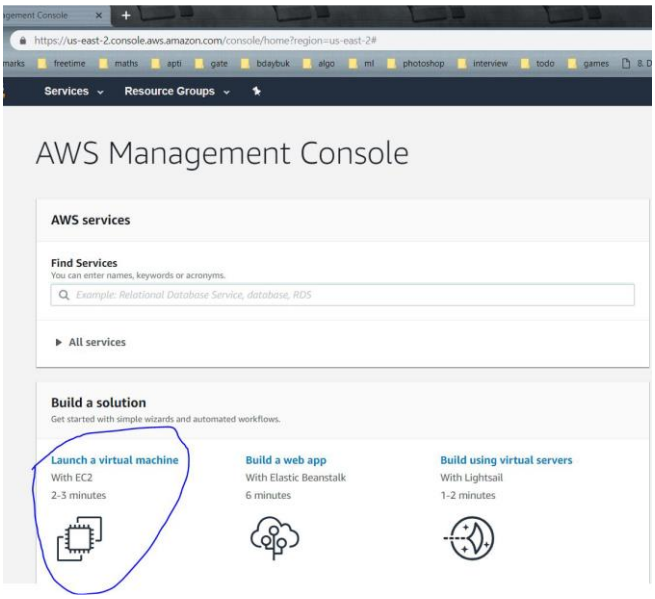
Packages needed:

1. pip3
2. pandas
3. numpy
4. sklearn
5. beautifulsoup4
6. lxml
7. flask
8. re

Launch a micro instance on AWS.

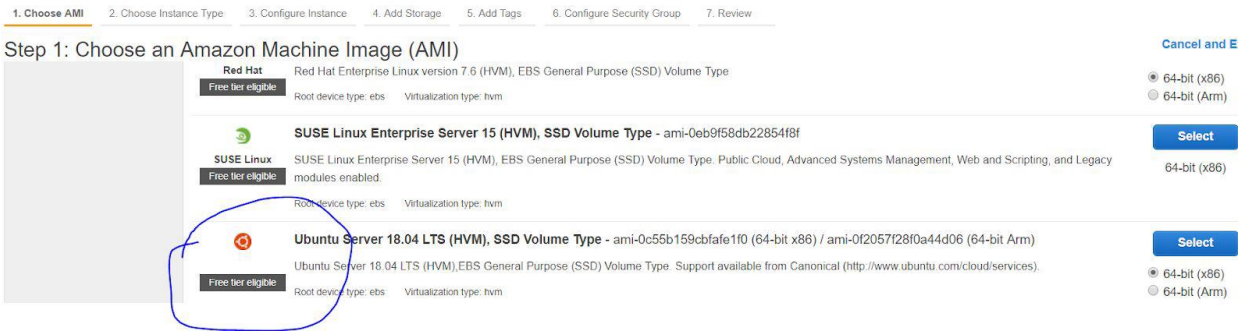
Creating an instance:

1. Create an AWS account https://aws.amazon.com , https://portal.aws.amazon.com/billing/signup#/start
2. Login: https://console.aws.amazon.com
   After login:

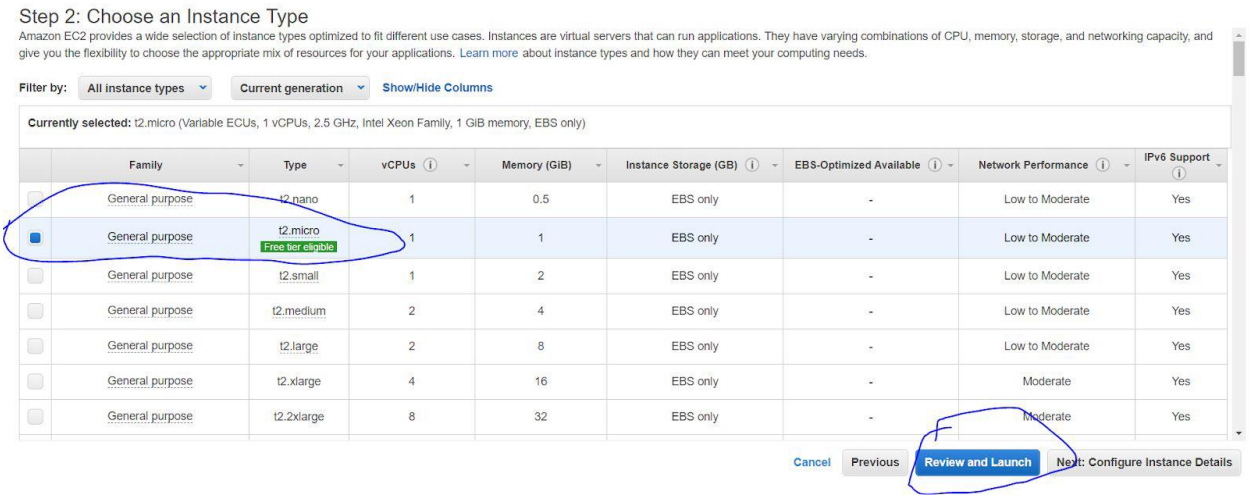


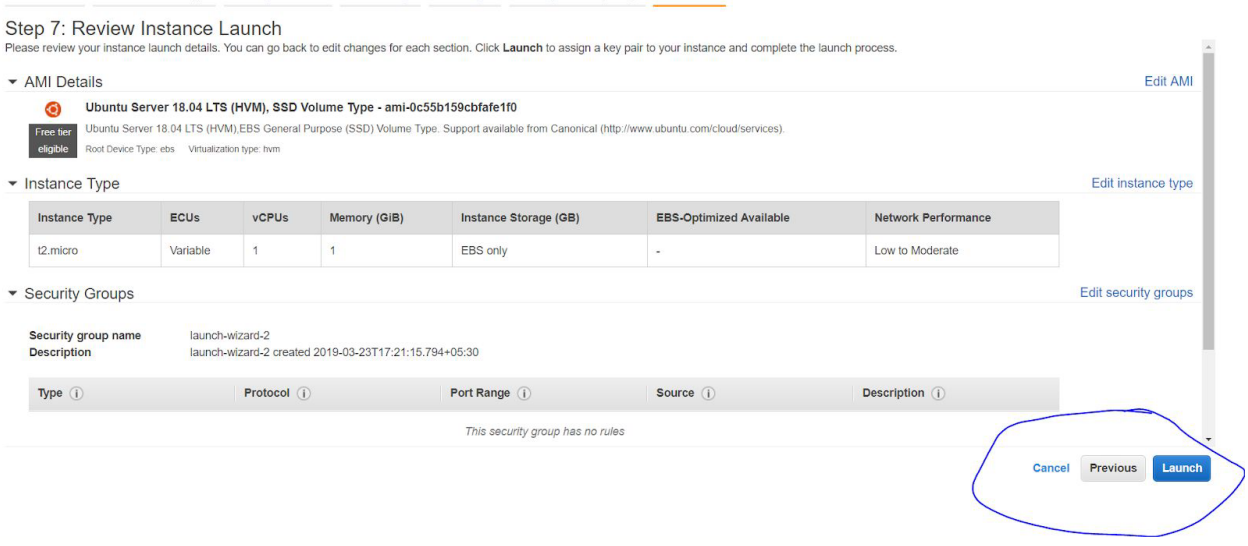
Launch the EC2 instance

3. Choose the ubuntu free tire

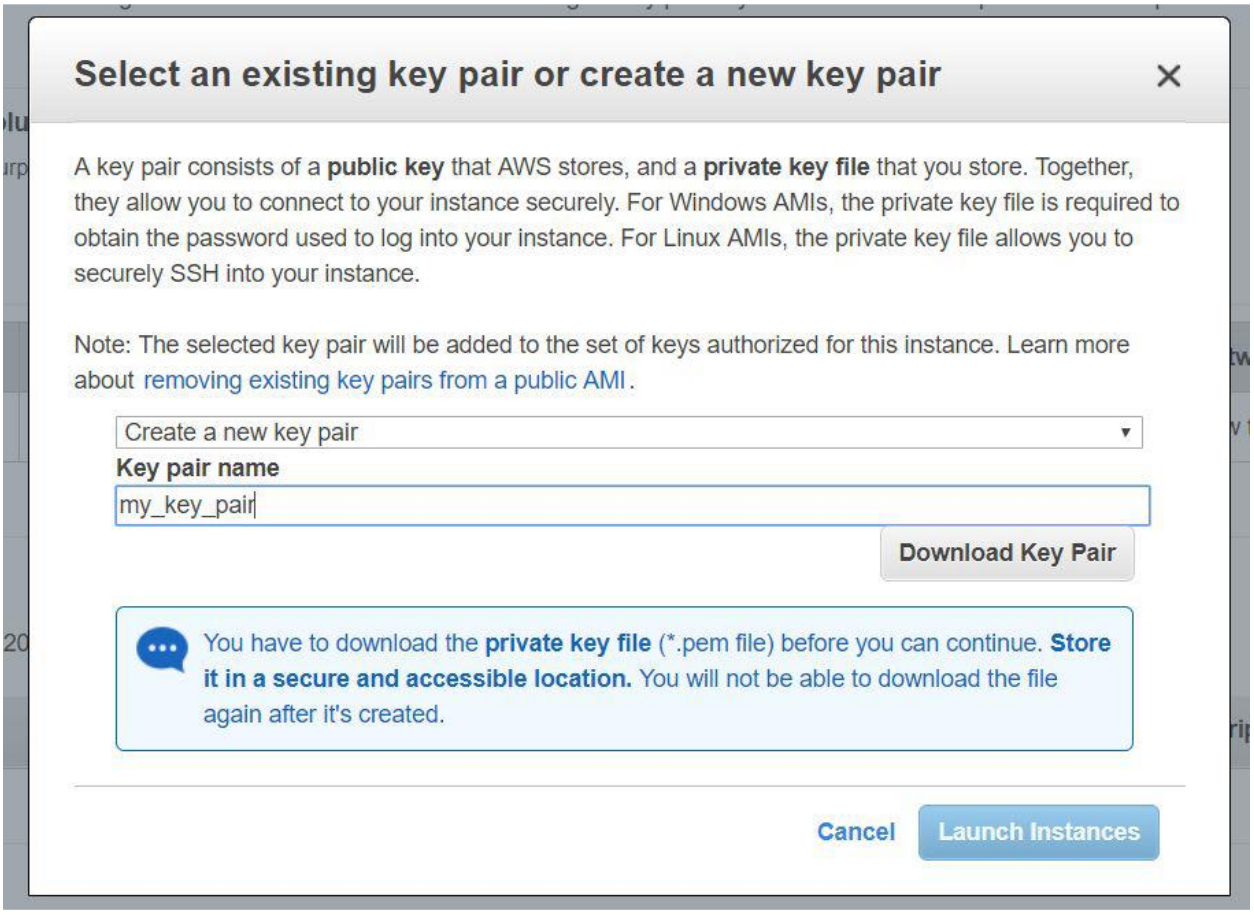


4. Choose t2.micro free tier eligible & then Click on review and launch

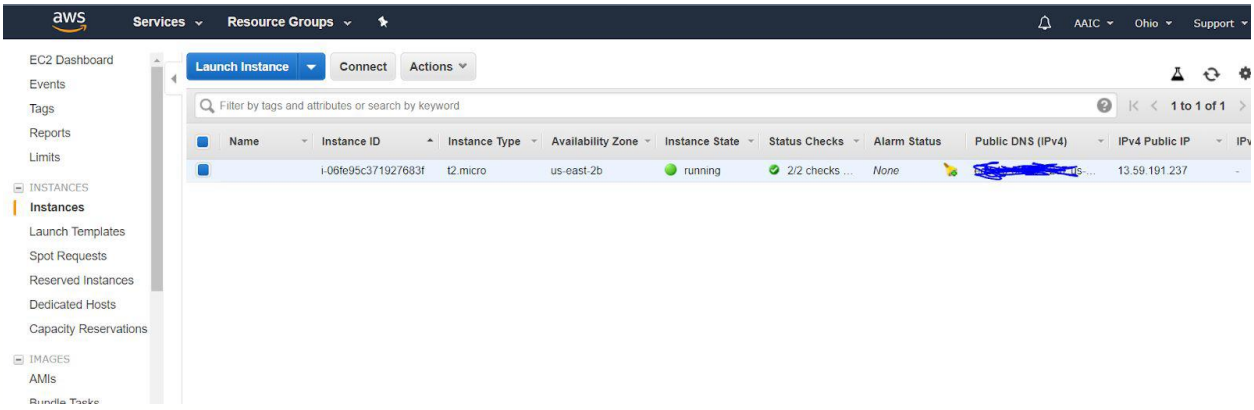




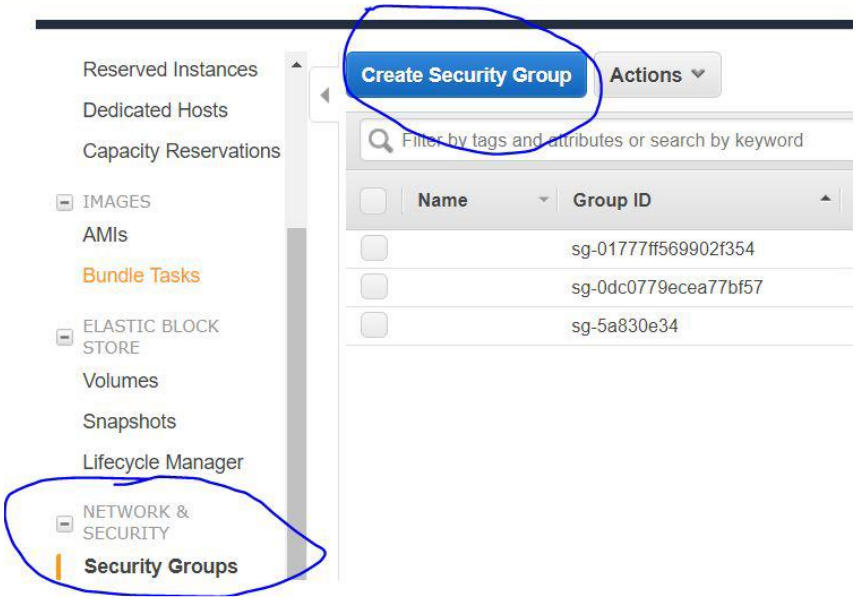
5. Click on “Download Key Pair” and save the .pem file then click on “Launch Instance”

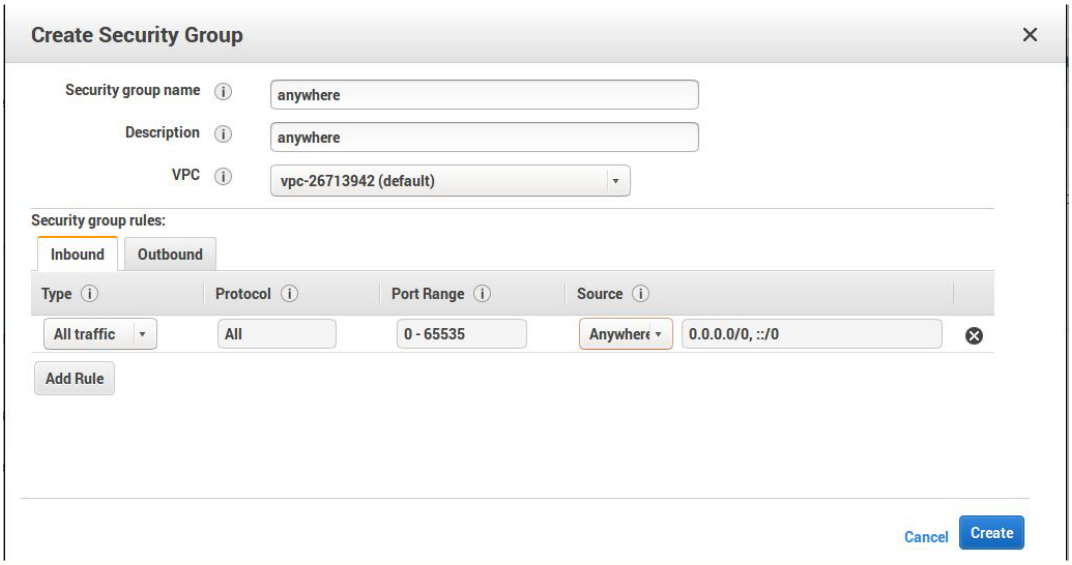


6. You will see this screen, you have successfully launched the an EC2 instance, now we need to
Launch an flask api in it



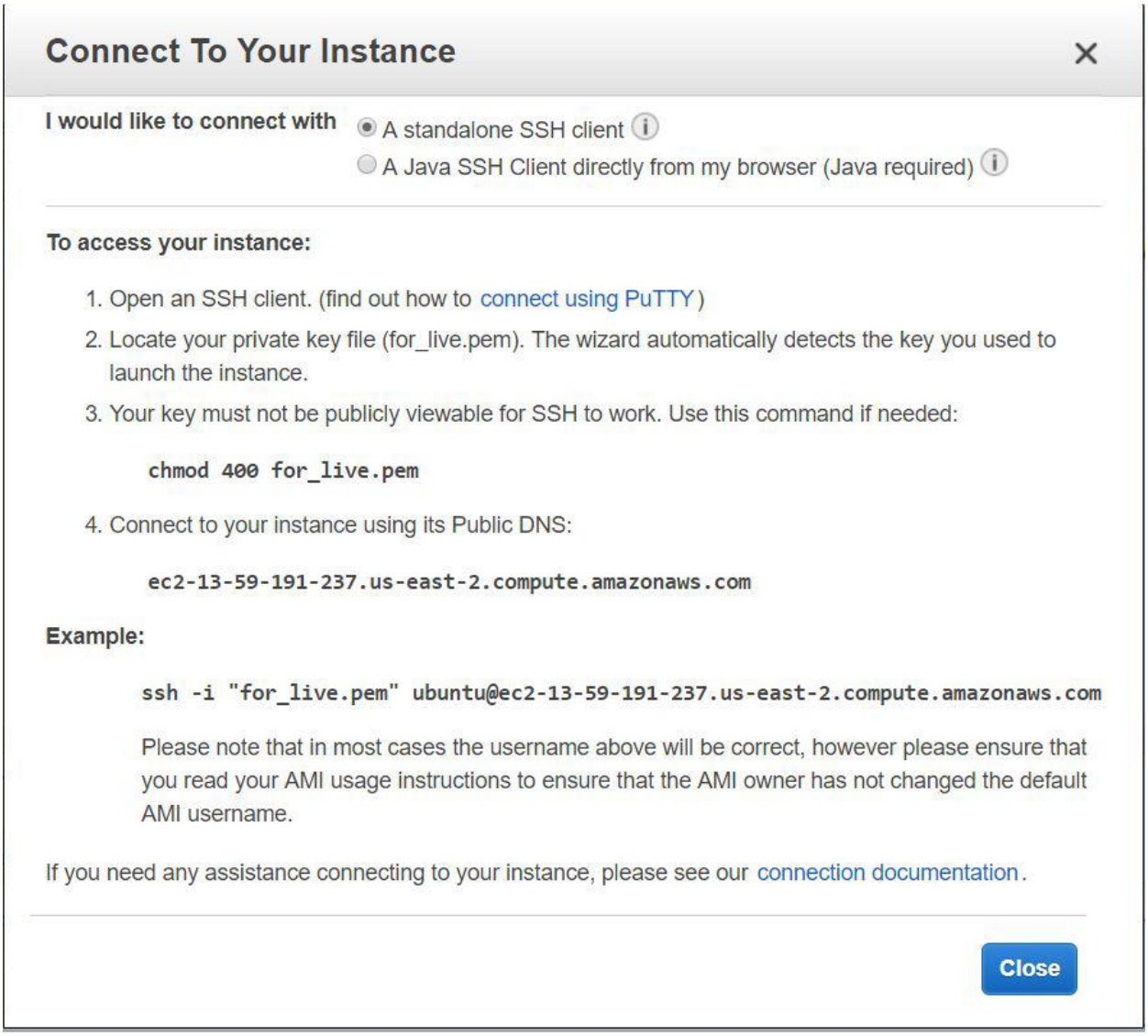
7. Select the “Network & security” -> Security groups and then click “Create Security Group”





Then add the specific security group to **network interface**

Connect to the AWS box



Move the files to an AWS EC2 instance/box

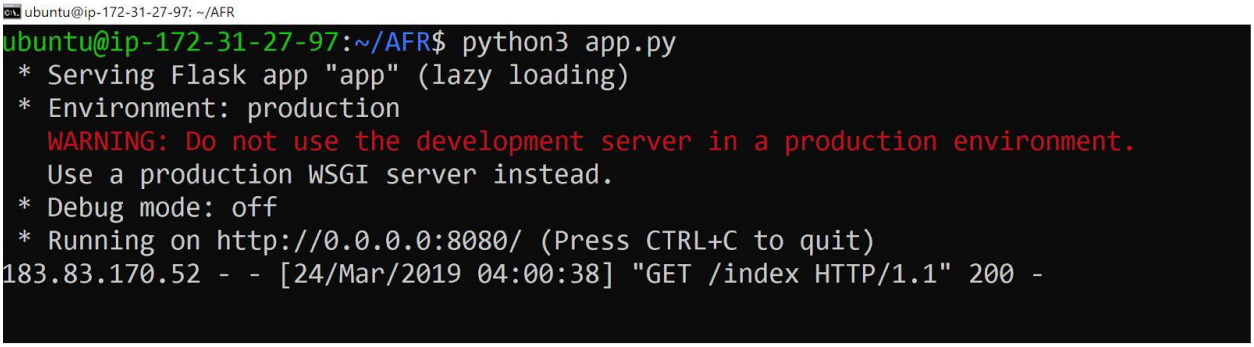
Command line to copy files

C:\Users\Asus\OneDrive\Desktop> **scp -r -i "for_live.pem" ./AFR**
ubuntu@ec2-13-59-191-237.us-east-2.compute.amazonaws.com :~/

Install all packages needed on the AWS box

sudo apt-get install python3-pip
pip3 install <each of the following packages>
Packages needed:
pip3
pandas
numpy
sklearn
beautifulsoup4
lxml
flask
re

Run app.py on the AWS box.



And then check in browser

For our case test link will be like –

ec2-18-224-96-3.us-east-2.compute.amazonaws.com:8080/querypage

- **Scalability, limitations, latency of our system**

In terms of scalability the model could have been designed better, at this stage predicting sales of future dates does not provide much accurate results, but in case of regression it is always an issue.

Whereas we have used free tier to deploy the project, so the latency is not good. But deploying at proper infrastructure it would have been results better.

Though we have not measure the latency.

- **Given more time**
  LSTM model could have been reengineered and used in production. Also standard Time series forecasting models like AR, MA, ARMA, ARIMA, SARIMA, SARIMAX, VAR, VARMA, VARMAX etc.

References

1. https://www.appliedroots.com/