ML Regression Analysis, Implemented on Streamlit/Python, Deployed on AWS EC2

In a previous article, I got quite satisfactory results using various machine learning regression algorithms in estimating the compressive strength values of concrete using 8 different parameters. I wrote a follow-up to this article and applied deep learning to the same data set and compared the performances.

In this article I am going to give the details of the steps involved in implementing a Machine Learning Regression Analysis on Streamlit, followed by deploying on AWS EC2.

Starting with definitions, Streamlit is an open-source Python library that makes it easy to generate and share beautiful, custom web apps for machine learning and data science. It is possible to build and deploy (locally) powerful data apps in just a few minutes.

Amazon Web Services (AWS) offers reliable, scalable, and inexpensive cloud computing services. The term cloud computing services is a broad category and it encompasses the IT resources provided over the internet. Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

I used a Kaggle dataset. As explained on the website, the first seven parameters are the addition to the concrete (units in kg in a m^3 mixture and "Age" in days (1 to 365)). There are 8 quantitative input variables, and 1 quantitative output (Compressive Strength (in MPa) of the concrete); 1030 instances with no missing data.

Data Set

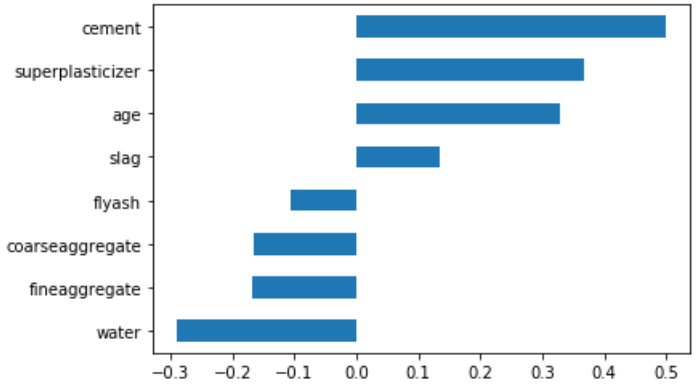
Pandas describe function gives basic statistical details of numerical columns. Minimum and maximum values will be important while getting input from the user in Streamlit.

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
cement	1030.0	281.167864	104.506364	102.00	192.375	272.900	350.000	540.0
slag	1030.0	73.895825	86.279342	0.00	0.000	22.000	142.950	359.4
flyash	1030.0	54.188350	63.997004	0.00	0.000	0.000	118.300	200.1
water	1030.0	181.567282	21.354219	121.80	164.900	185.000	192.000	247.0
superplasticizer	1030.0	6.204660	5.973841	0.00	0.000	6.400	10.200	32.2
coarseaggregate	1030.0	972.918932	77.753954	801.00	932.000	968.000	1029.400	1145.0
fineaggregate	1030.0	773.580485	80.175980	594.00	730.950	779.500	824.000	992.6
age	1030.0	45.662136	63.169912	1.00	7.000	28.000	56.000	365.0
csMPa	1030.0	35.817961	16.705742	2.33	23.710	34.445	46.135	82.6

I checked the correlation of all the parameters (columns) and noticed that cement has the highest positive correlation while water has the maximum negative correlation. Flyash has the minimum effect on the compressive strength of the concrete.

```
df.corr()['csMPa'].sort_values().head(11)[:-1].plot.barh();
```



Machine Learning

In the original paper, I applied five different algorithms, but here I will use only XGBoost, which gave the highest r2-score value. Remember, the target here is to deploy the analysis by Streamlit on AWS EC2.

XGBOOST Regressor

```
X=df.drop(['csMPa'], axis=1)
y=df['csMPa']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

```
import xgboost
```

```
model_xg = xgboost.XGBRegressor()
model_xg.fit(X_train, y_train)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.300000012, max_delta_step=0, max_depth=6,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=100, n_jobs=0, num_parallel_tree=1,
             objective='reg:squarederror', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

```
y_pred = model_xg.predict(X_test)
```

```
eval_metrics(y_test, y_pred)
```

```
r2_score: 0.9300780437301508
 mae: 2.841180106690786
 mse: 18.41601985920957
 rmse: 4.291389036105858
```

```
import pickle
pickle.dump(model_xg, open('model_xg', 'wb'))
```

At the last step, I imported ***pickle*** and saved the XGBoost model that I used for the analysis. Python's pickle module is very useful when you're working with machine learning algorithms, where you want to save them to be able to make new predictions at a later time, without having to rewrite everything or train the model all over again.

Streamlit

Now that we have the model saved as a .pkl file, we can start preparing the streamlit part. You can use any word editor, but you must save the file with .py extension.

Below, you can see that first steps are importing the libraries, inserting a photo and a title to the app.

jupyter Streamlit_Concrete.py

File Edit View Language

```
# Import the Libraries

import streamlit as st
import pickle
import pandas as pd

# Nice Concrete Photo

from PIL import Image
im = Image.open("taylor-smith-PJMKu7RQNJ8-unsplash.jpg")
st.image(im, width = 700, caption = "by Taylor Smith, unsplash.com")

# App Title

html_temp = """
<div style="background-color:blue;padding:1.5px">
<h1 style="color:white;text-align:center;">Compresssive Strength Prediction </h1>
</div><br>"""
st.markdown(html_temp,unsafe_allow_html=True)
```

Next step will be getting the input values from the user. The left-most part of the screen will be reserved for the user to enter the values (all numeric). Using those values, a dictionary named my_dict will be formed and this dictionary will be converted to a Pandas DataFrame.

```
# Get the Input Values From Sidebar

st.sidebar.title('Please Enter the Following Parameters')

cement=st.sidebar.slider("Amount of Cement", 102, 540, step=5)
slag=st.sidebar.slider("Amount of Slag", 0,360, step=5)
flyash=st.sidebar.slider("Amount of Flyash", 0,200, step=1)
water=st.sidebar.slider("Amount of Water", 120, 250, step=1)
superplasticizer=st.sidebar.slider("Amount of Superplasticizer", 0, 32, step=1)
coarseaggregate=st.sidebar.slider("Amount of Coarseaggregate", 800, 1145, step=1)
fineaggregate=st.sidebar.slider("Amount of Fineaggregate", 594, 993, step=1)
age=st.sidebar.slider("Age in Days", 1, 365, step=1)

# Prepare a Python Dictionary Using the Input Values and Convert to a DataFrame

def csMPa():
    my_dict = {"cement" :cement,
        "slag":slag,
        "flyash": flyash,
        "water": water ,
        "superplasticizer": superplasticizer,
        "coarseaggregate": coarseaggregate,
        "fineaggregate": fineaggregate,
        "age":age}

    df_sample = pd.DataFrame.from_dict([my_dict])
    return df_sample
```

Final steps will be opening and reading the XGBoost model prepared and saved before in the main Python file. The model will make a prediction and display the predicted compressive strength value.

```
dfc = csMPa()

model = pickle.load(open("model_xg", "rb"))

if st.sidebar.button("Submit"):
    result = (model.predict(dfc))
    st.success(f"Compressive Strength Prediction of the Concrete is {result} MPa")
```

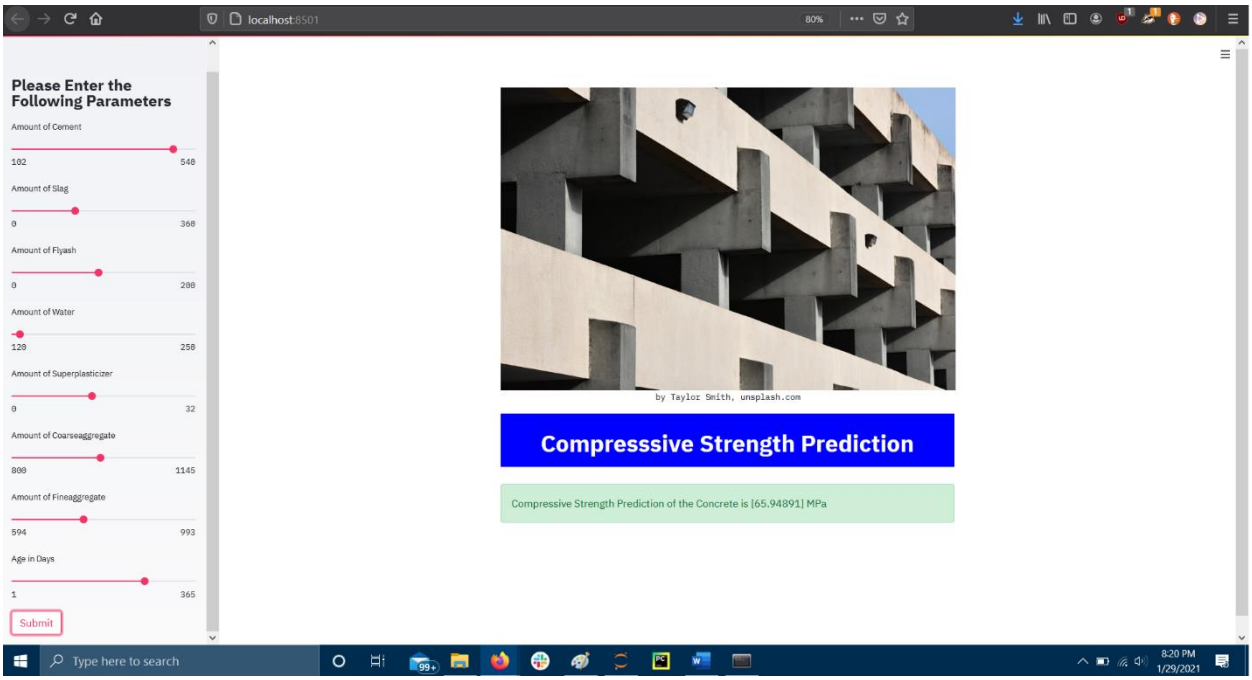
Next stage will be saving and running the code in Anaconda or a similar simple bash shell terminal. Follow the following steps:

```
pip install streamlit
streamlit run YOURCODE.py
```

Just make sure that all the files that are needed are in the same folder. In this case, you will need these files to run the app successfully:

- Streamlit file saved with a .py extension.
- The model saved as a pickle file.
- The photo (not always needed, I used it , so I need it).

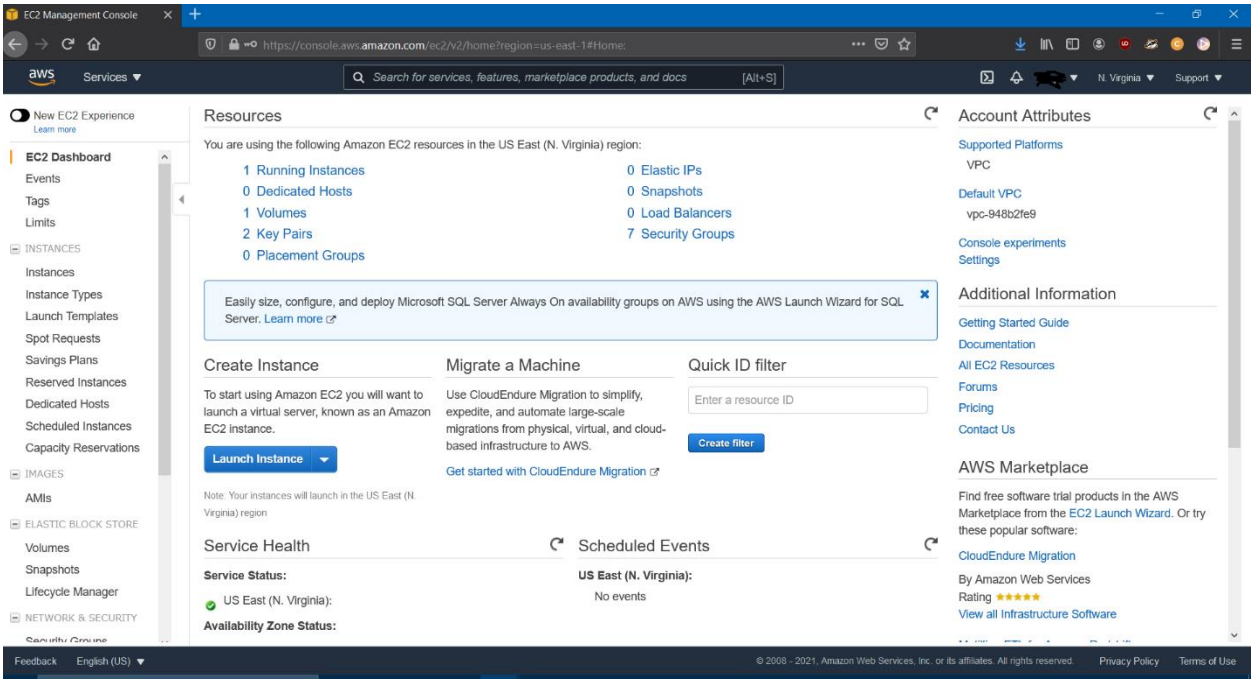
Once you run the code, you will get the screen below:



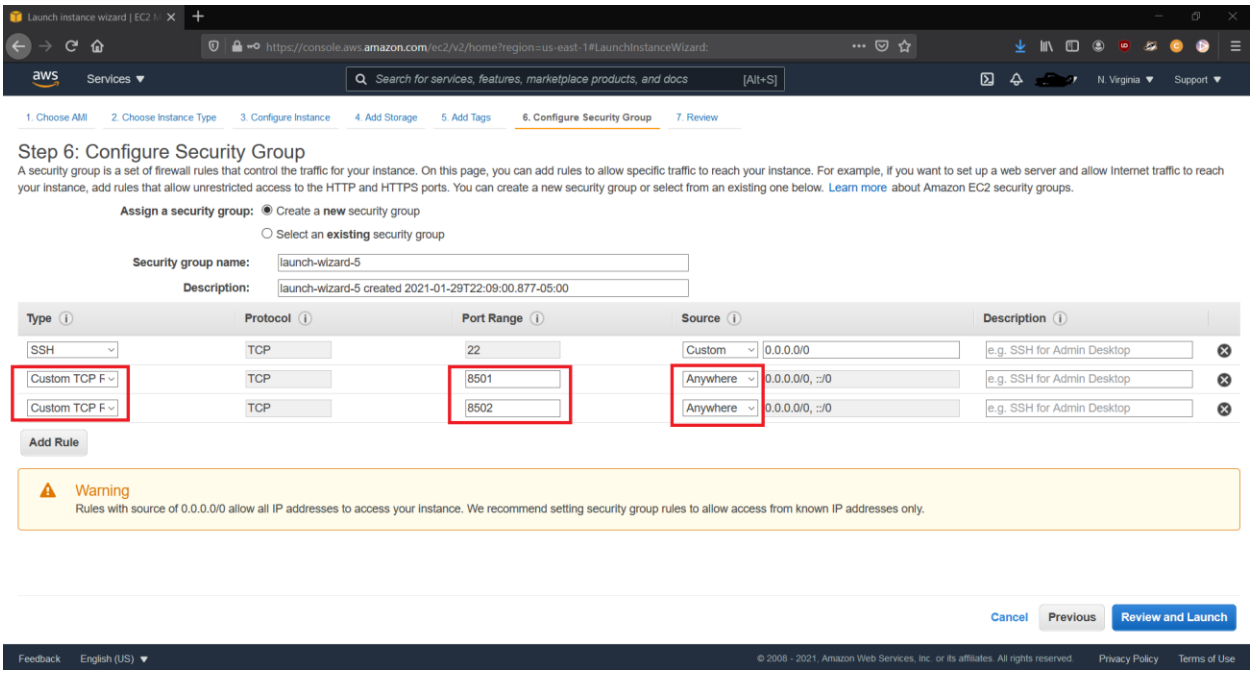
Your app is running, but only locally. If you want to make your app accessible from other computers, AWS deployment may be a good alternative.

AWS EC2 Deployment

First, you must open an AWS account and sign into it. Then, you need to go to EC2 page and launch an instance as shown below:



You will need a basic computer for this application, so go for a free configuration and take the default setup until the 6th stage. Here add two rules (Custom TCP), label the Port Ranges 8501 and 8502, turn Sources to Anywhere (all marked in red).



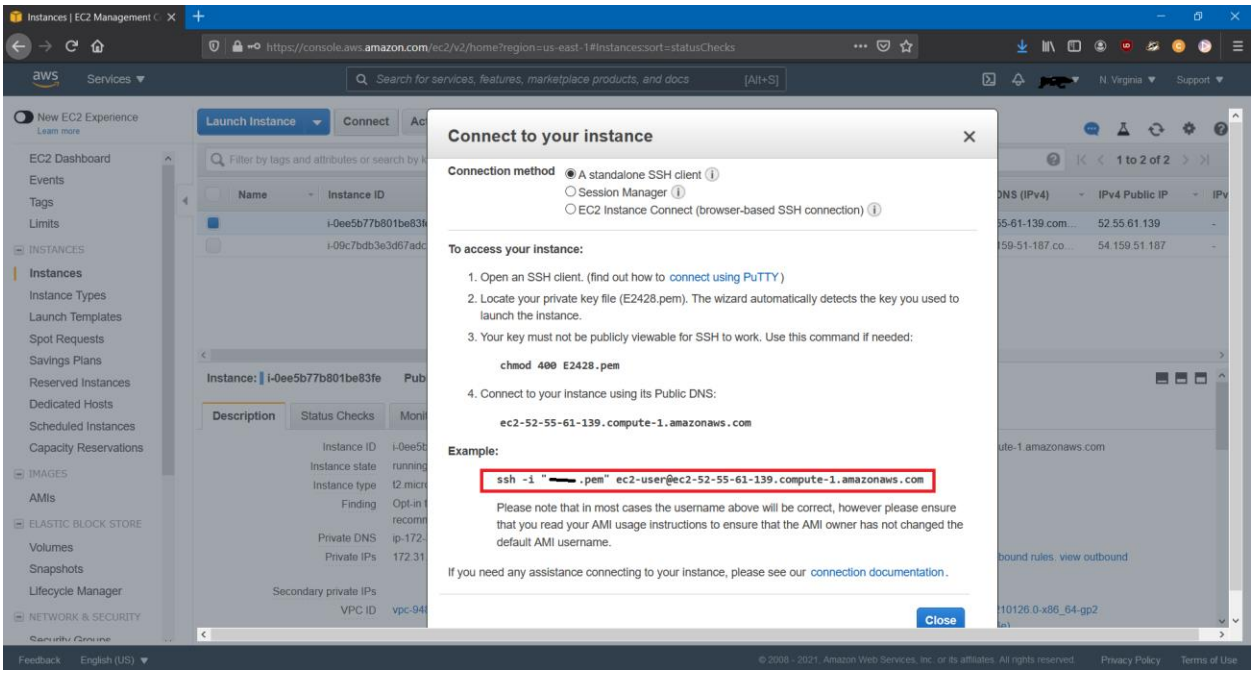
Then go to “Review and Launch” step, check everything and launch the system. You have to use a key pair here. Save this file to the folder where all the streamlit files are located.

Follow the following steps:

- Open Git Bash in the above-mentioned file folder,
- Use the following command to reduce the permission level of your pem file from ***w*** to ***r***:

```
chmod 400 yourpemname.pem
```

- Connect to the instance by running the ssh commands (shown in red) on Git Bash:



Once you are connected, install Python 3.7 (or the version that you will be using) and git to EC2.

If you want to keep the app running on EC2, make a repo on GitHub and save the files there. Copy the https address of this repo.

Now, clone your repo to EC2:

```
git clone https://github.com/sxxxxxxxxxx/streamlit.git
```

To check the success of the cloning, ls and then get into that folder.

Next step will be to make a virtual environment and activating it:

```
python3 -m venv myproject
source myproject/bin/activate
```

Next, install the packages required by the Streamlit. In my case the following were installed:

```
pip install sklearn
pip install xgboost
pip install streamlit
```

Once you installed all the necessary packages, you can run the app:

```
streamlit run app.py
```

You will get the following message in the terminal and you will be able to view your Streamlit app in your browser.

```
Network URL: http://172.31.28.28:8501
External URL: http://18.188.133.122:8501
```

To keep the app running, you need to install tmux and generate a new tmux session and run the app again.

```
tmux new -s st_instance
```

```
streamlit run app.py
```

Even if you close the Git Bash, the app will continue to work so long the EC2 is running.

Conclusion

In this article I gave the details of the steps involved in implementing a Machine Learning Regression Analysis on Streamlit, followed by deploying on AWS EC2.

You can access to this article and similar ones here.

Photo by Nathan Anderson on Unsplash