**A Project Report on**

**Predicting the energy output of wind turbine**

**based on weather condition**

Submitted in partial fulfillment for the award of

**CALL FOR CODE 2020**

**BY**

**TEAM NAME: CRAZY BRAINS**

**TEAM LEADER: PRABHANJAN KUMAR**

**TEAM SIZE: 1**

**TEAM MEAMBERS:**

**1.PRABHANJAN KUMAR**

**DECLARATION**

We declare that this project work is composed by overselves, that the work contained here in our own except where explicit stated otherwise in the text ,and that this work has not been submitted for any other degree or professional qualification except as specified

**TEAM NAME: CRAZYBRAINS**

**ACKNOWLEDGEMENT**

The project report entitled “**Predicting the energy output of wind turbine based on weather condition”** has been carried out as a part of **CALL FOR CODE 2020**

We express our sincere thanks to our mentors Gayatri and Hemanth Kumar, who helped us by giving their valuable Suggestions and encouragement

We would like to take opportunity to thank to **Smart Bridge** for Giving us the opportunity for doing the project

**TEAM NAME: CRAZYBRAINS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1** | |  | **INTRODUCTION** | |  |
|  | |  | 1.1 Overview | |  |
|  | |  | 1.2 Purpose | |  |
| **2** | |  | **LITERATURE SURVEY** | |  |
|  | |  | 2.1 Existing problem | | |
|  | |  | 2.2 Proposed solution | | |
| **3** | |  | **THEORITICAL ANALYSIS** | |  |
|  | |  | 3.1 Block diagram | |  |
|  | |  | 3.2 Hardware / Software designing | | |
| **4** | |  | **EXPERIMENTAL INVESTIGATIONS** | | |
| **5** | |  | **FLOWCHART** | |  |
| **6** | |  | **RESULT** |  |  |
| **7** | |  | **ADVANTAGES & DISADVANTAGES** | | |
| **8** | |  | **APPLICATIONS** | |  |
| **9** | |  | **CONCLUSION** | |  |
| **10** | |  | **FUTURE SCOPE** | | **`** |
| **11** | |  | **BIBILOGRAPHY** | |  |
|  | |  | **APPENDIX** | |  |
|  |  | | A. Source code | |  |

**INTRODUCTION**

* 1. **Overview:**

Time Series analysis comprises methods for analyzing time series data in order to extract meaningful statics and other characteristics of the data. Timeseries forecasting is the use of a model to predict future values based on previously observed values. Time series are widley used for non-stationary data, like economic, wether, stock price and retail sales. The aim here is to develop a model that can accurately predict the power output of wind form based on weather conditions in the site .The following Steps are involved in building model

* Data Collection
* Data Visualization
* Data Preprocessing
* Splitting data into train, test sets
* Model building
* Making predictions from the model
  1. **Purpose:**

Wind energy plays an increasing role in the supply of energy world-wide. The energy output of a wind farm is highly dependent on the wind conditions present at its site. If the output can be predicted more accurately, energy suppliers can coordinate the collaborative production of different energy sources more efficiently to avoid costly overproduction.

**LITERATURE SURVEY**

**2.1 Existing problem:**

The Existing Predicting the energy output of wind turbine based on weather condition applications are more complex and not updated with the current data of the weather conditions so we need to update the current data with the existing old data and it is difficult to analize the application.

**2.2 Proposed solution:**

We develop a model by time series forecasting because the data is Sequential data . LSTM's are the best for the Sequential data it gives better results for Sequential data.it makes development easily reduce the complexity in the code . we train the model with last 5 days of data to predict the power in 72 hours future.

We develop the User Interface using the Django frame work it makes very easy to create User Interface the power predicted is showed in the Graph using Plotly .

**THEORITICAL ANALYSIS**

**3.1 Block diagram:**

Start

Enter data

valid

Load model

Predict output

Plot results

Stop

**3.2 Hardware / Software designing:**

**Technologies and tools:**

**Software Requirments :**

**→**Languages :Python

**→**Packages :Tensorflow, numpy, pandas, matplotlib

**→**Tools :IBM Watson Studio, IBM Cloud for Deployment

**Hardware Requirments:**

**→** GPU :4GB(NVIDIA-CUDA Compatible)

**→** RAM :4GB

**EXPERIMENTAL INVESTIGATIONS**

**Exploring IBM Cloud Services:**

IBM provides many services some of them are

**Machine Learning:** IBM Watson Machine Learning  make smarter decisions,slove through problems, and improve user outcomes

**Watson Studio:** Embed AI and Machine learning into Business,create custom models using our own data

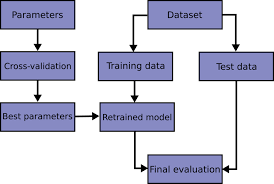
**Exploring IBM Watson Services:**

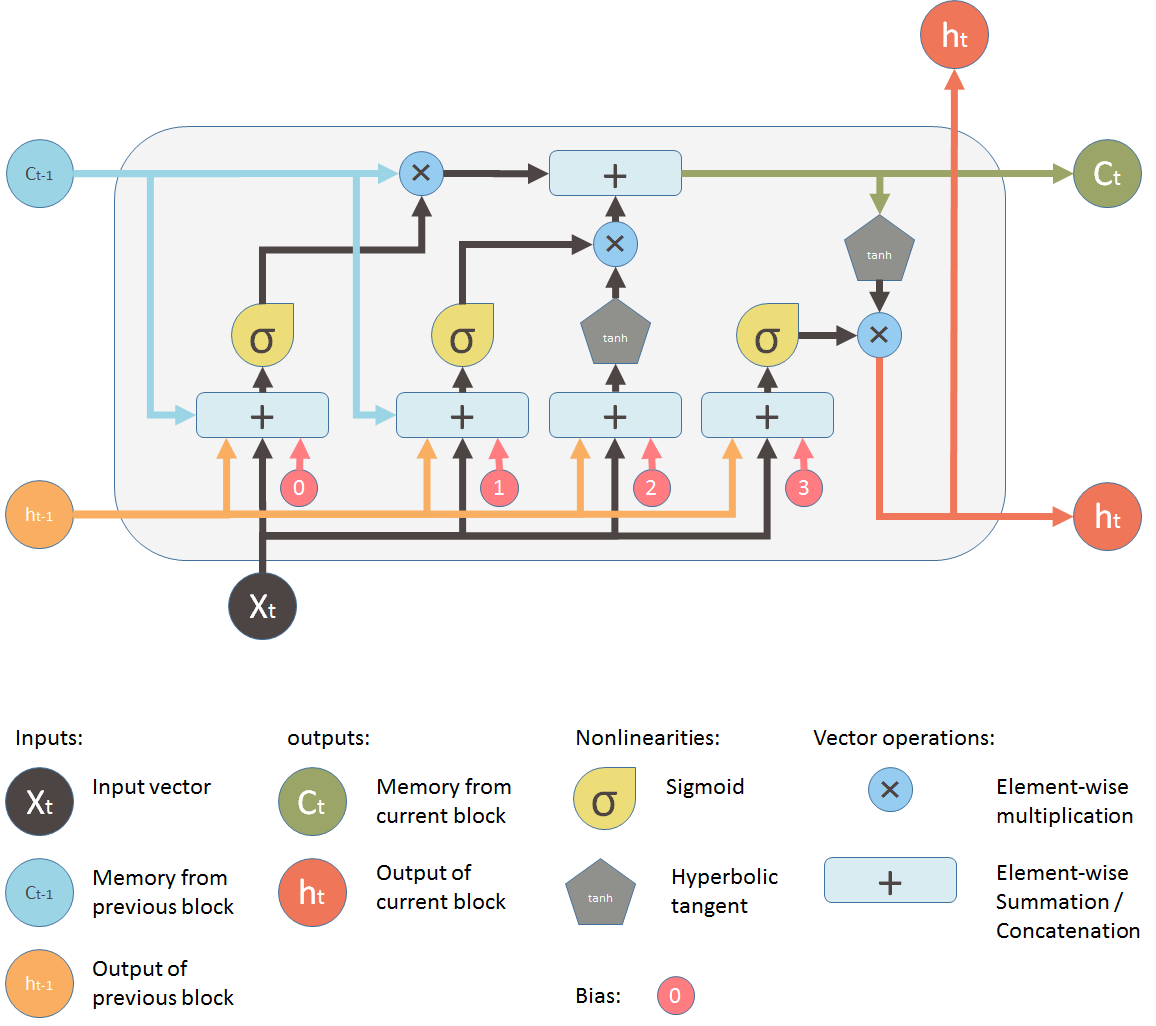
Watson Studio provides you with the environment and tools to solve your business problems by collaboratively working with data. You can choose the tools you need to analyze and visualize data, to cleanse and shape data, to ingest streaming data, or to create and train machine learning models.

**Watson Machine Learning:**Using IBM Watson Machine Learning, you can build analytical models and neural networks, trained with your own data, that you can deploy for use in applications.Watson Machine Learning provides a full range of tools and services so you can build, train, and deploy Machine Learning models. Choose from tools that fully automate the training process for rapid prototyping to tools that give you complete control to create a model that matches your needs.

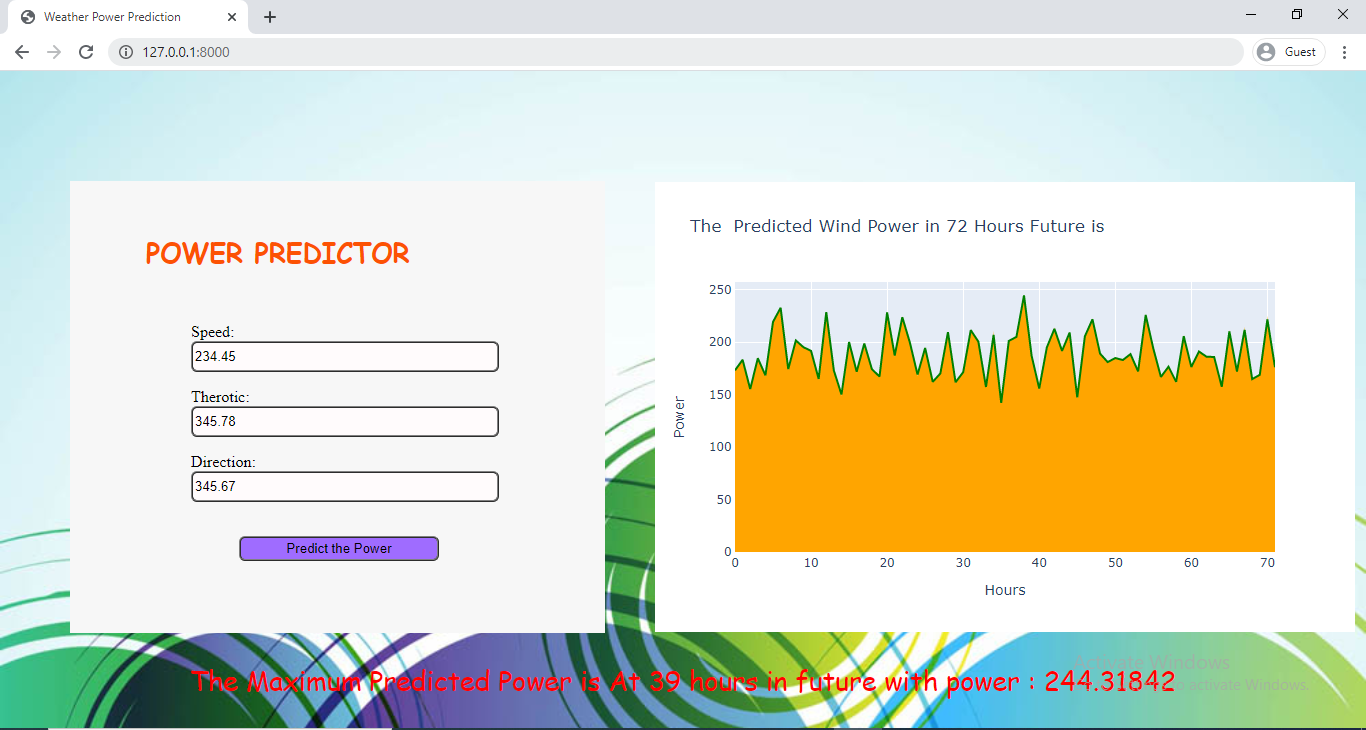
**Watson Notebook:**Watson Notebook provide an interactive programming environment for working with data, testing models, and rapid prototyping.

**FLOWCHART**

****

****

**RESULT**

****

**ADVANTAGES & DISADVANTAGES**

**ADVANTAGES:**

* Predict the results accuretley
* Easy to understand the visualization of the results
* Easy to intract with the model

**DISADVANTAGES:**

* Lack of data to train the model
* Requires GPU for training Speed

**APPLICATIONS**

Energy suppliers can coordinate the collaborative production of different energy sources more efficiently to avoid costly overproduction. It is used to reduce the cost of production.

**CONCLUSION**

This Project give you knowledge of time series forecasting and shows you to how to work with Sequential data and how LSTM’s Solve the Sequential Problems and This Project Gives knowledge about how to save model and used for the later use and get the knowledge on django applications.

**FUTURE SCOPE**

This Project helps to understand the amont of Power Generated in the future and helps user to intract easily and reduce the cost of Production.

**BIBILOGRAPHY**

**Name:** Prabhanjan Kumar

**College Name:** Bapatla Engineering College

**Work Title:** Predicting the energy output of wind turbine based on weather condition

**APPENDIX**

**Code:**

**Description about data:**

This data is in csv format containing 50530 rows and 5 coloums.The dataset contain 4 features they are Wind Speed (m/s),LV ActivePower (kW),Theoretical\_Power\_Curve (KWh) and Wind Direction (Â°) with Date/Time. These were collected every 10 minutes that means for single hour we have 6 observations . A single day contain 144(6\*24). This [data](https://www.kaggle.com/berkerisen/wind-turbine-scada-dataset) was Taken From wind Turbine scada system that is working and generating power in Turkeyobservations

**Importing Libraries:**

**tensorflow:** TensorFlow bundles together a slew of machine learning and deep learning  models and algorithms and makes them useful  it helps to build models easily.

**pandas:** Used to load the data,The dataset is in csv format . pandas is used to read csv data.

**numpy:**NumPy is a general-purpose array-processing package.It provides a high-performance multidimensional array object, and tools for working with these arrays.

**matplotlib:**it is used to visualization of data. visualizing data is important to understand easily.

**Code:**

import tensorflow as tf

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import seaborn as sns

from sklearn.preprocessing import MinMaxScaler

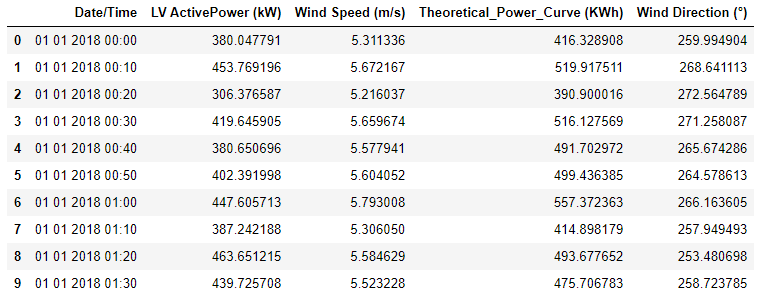
from tensorflow.keras.layers import Dense,LSTM,Dropout

from tensorflow.keras.models import Sequential

**Loding Data:**

dataset = pd.read\_csv(body)

dataset.head(10)

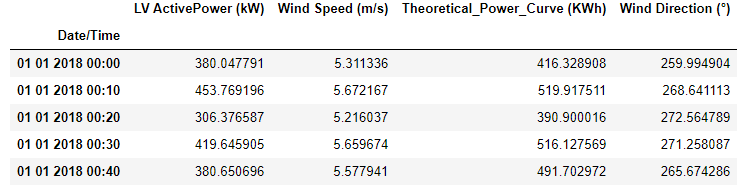


features\_intake = ['LV ActivePower (kW)','Wind Speed (m/s)','Theoretical\_Power\_Curve (KWh)','Wind Direction (°)']

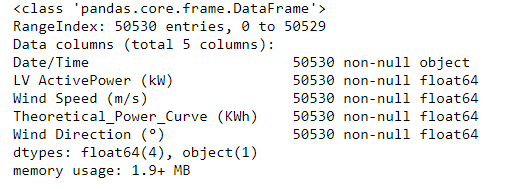
features = dataset[features\_intake]

features.index = dataset['Date/Time']

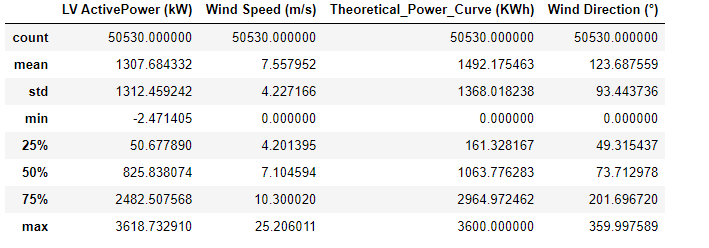
features.head()



dataset.info()

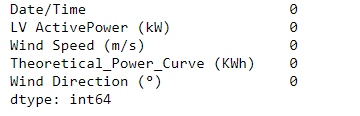


dataset.describe()



**Checking for null values:**

dataset.isnull().sum()

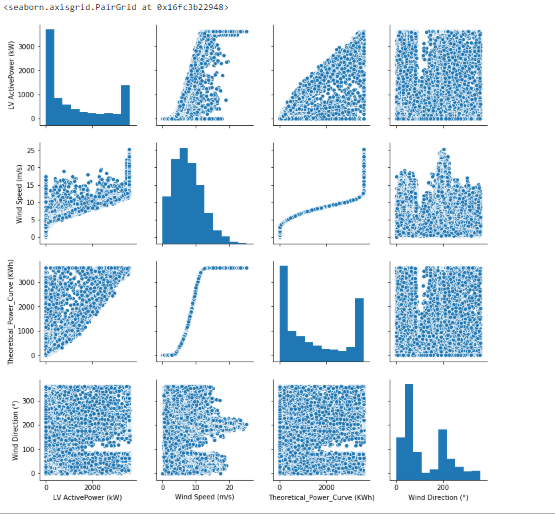


**Data visualization:**

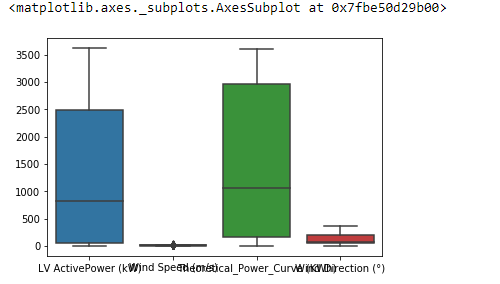
features.plot(subplots=True)



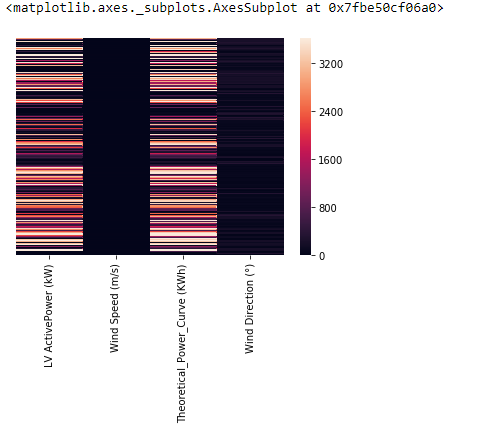
sns.pairplot(dataset[['LV ActivePower (kW)','Wind Speed (m/s)','Theoretical\_Power\_Curve (KWh)','Wind Direction (°)']],diag\_kind="auto")



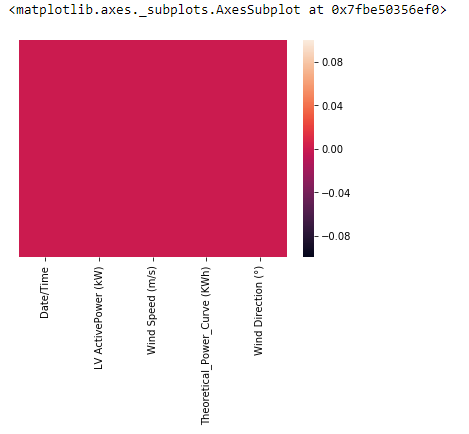
sns.boxplot(data=dataset)



sns.heatmap(dataset[['LV ActivePower (kW)','Wind Speed (m/s)','Theoretical\_Power\_Curve (KWh)','Wind Direction (°)']],yticklabels=False)



sns.heatmap(dataset.isnull(),yticklabels=False)



**Data Preprocessing:**

**Normalization :**

 The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges.

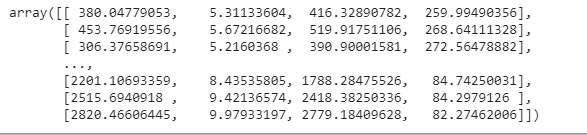
**MinMaxScaler :**

For each value in a feature, MinMaxScaler subtracts the minimum value in the feature and then divides by the range. The range is the difference between the original maximum and original minimum. it is the simplest method and consists in rescaling the range of features to scale the range in [0, 1] or [−1, 1].

scale=MinMaxScaler(feature\_range=(0,1))

dataset = features.values

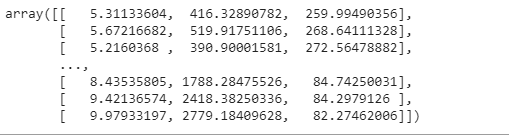
dataset



data\_in=dataset[:,1:4]

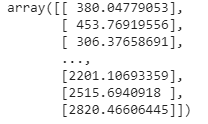
data\_out=dataset[:,0]

data\_in



data\_out=np.reshape(data\_out,(-1, 1))

data\_out



data\_in=scale.fit\_transform(data\_in)

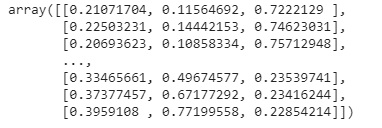
data\_out=scale.fit\_transform(data\_out)

data\_out=np.reshape(data\_out,(data\_out.shape[0],))

data\_out

sns4.PNG

data\_in



def multivariate\_data(dataset, target, start\_index, end\_index, history\_size,

                      target\_size, step, single\_step=False):

    data = []

    labels = []

    start\_index = start\_index + history\_size

    if end\_index is None:

        end\_index = len(dataset) - target\_size

    for i in range(start\_index, end\_index):

        indices = range(i-history\_size, i, step)

        data.append(dataset[indices])

        if single\_step:

            labels.append(target[i+target\_size])

        else:

            labels.append(target[i:i+target\_size])

  return np.array(data), np.array(labels)

def create\_time\_steps(length):

    return list(range(-length, 0))

def show\_plot(plot\_data, delta, title):

    labels = ['History', 'True Future', 'Model Prediction']

    marker = ['.-', 'rx', 'go']

    time\_steps = create\_time\_steps(plot\_data[0].shape[0])

    if delta:

        future = delta

    else:

        future = 0

    plt.title(title)

    for i, x in enumerate(plot\_data):

        if i:

            plt.plot(future, plot\_data[i], marker[i], markersize=10,

               label=labels[i])

        else:

            plt.plot(time\_steps, plot\_data[i].flatten(), marker[i], label=labels[i])

    plt.legend()

    plt.xlim([time\_steps[0], (future+5)\*2])

    plt.xlabel('Time-Step')

    return plt

past\_history = 720

future\_target = 432

step=6

train\_split=40000

we need to train the model based on our requirment to predict.to make prediction we choose to use 5 days of observation so we take 720(5\*144) observations to train the model.we need to predict the wind power in 72 hours future so we need to set future target as 432 (6\*72)

x\_train\_single, y\_train\_single = multivariate\_data(data\_in, data\_out, 0, train\_split , past\_history, future\_target, step,single\_step=True)

x\_val\_single, y\_val\_single = multivariate\_data(data\_in, data\_out,train\_split,None ,past\_history,

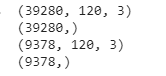
            future\_target,step,single\_step=True)

print(x\_train\_single.shape)

print(y\_train\_single.shape)

print(x\_val\_single.shape)

print(y\_val\_single.shape)



print ('Single window of past history : {}'.format(x\_train\_single[0].shape))

sns7.PNG

batch\_size=64

buffer\_size=len(x\_train\_single)

train\_data\_single = tf.data.Dataset.from\_tensor\_slices((x\_train\_single, y\_train\_single))

train\_data\_single = train\_data\_single.cache().shuffle(buffer\_size).batch(batch\_size).repeat()

val\_data\_single = tf.data.Dataset.from\_tensor\_slices((x\_val\_single, y\_val\_single))

val\_data\_single = val\_data\_single.batch(batch\_size).repeat()

for i,j in train\_data\_single.take(1):

    print(i.numpy().shape)

sns8.PNG

we develop a model by time series forecasting because the data is Sequential data . LSTM's are the best for the Sequential data it gives better results for Sequential data.

LSTM's process a time series step-by-step maintaining a internal state summarizing the information they have seen so for.

**single step model:**In a single step the model learns to predict a single point in the future based on some history provided.To reduce overfitting we use Dropout layers

model =Sequential()

model.add(tf.keras.layers.LSTM(120,input\_shape=x\_train\_single.shape[-2:],return\_sequences=True))

model.add(Dropout(0.25))

model.add(LSTM(72, activation='relu',return\_sequences=True))

model.add(Dropout(0.35))

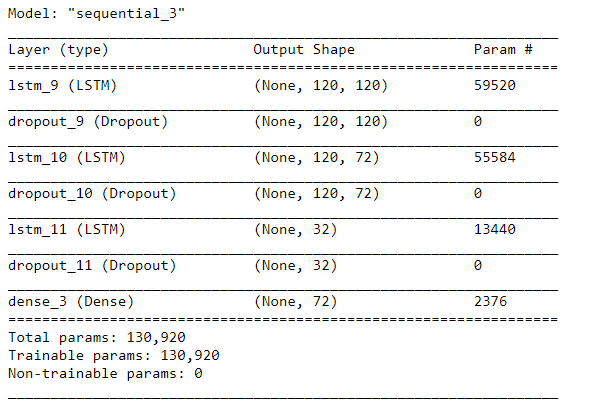
model.add(LSTM(32, activation='relu'))

model.add(Dropout(0.3))

model.add(Dense(72,activation='linear'))

model.compile(optimizer=tf.keras.optimizers.Adam(lr=3e-4), loss='mae')

model.summary()

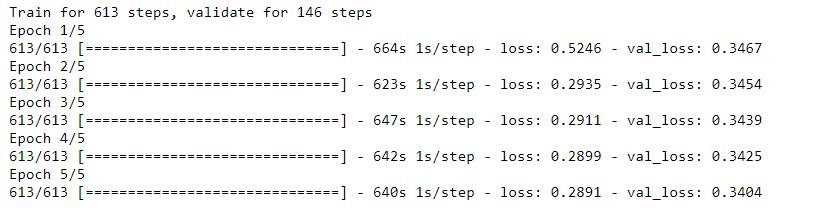


tf.random.set\_seed(13)

history = model.fit(train\_data\_single, epochs=5,

                                            steps\_per\_epoch=buffer\_size//batch\_size                                            validation\_data=val\_data\_single,

                                            validation\_steps=len(x\_val\_single)//batch\_size)



**Loss&optimizer:**

we use mean absolute error as loss and adam optimizer

**mean absolute error** (**MAE**) is a measure of [errors](https://en.wikipedia.org/wiki/Error_(statistics)) between paired observations expressing the same phenomenon.MAE is caluclated as it is an arithmetic average of the absolute errors.

**Adam** is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.

def plot\_train\_history(history, title):

    loss = history.history['loss']

    val\_loss = history.history['val\_loss']

    epochs = range(len(loss))

    plt.figure()

    plt.plot(epochs, loss, 'b', label='Training loss')

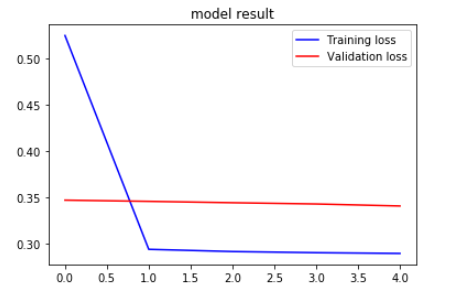
    plt.plot(epochs, val\_loss, 'r', label='Validation loss')

    plt.title(title)

    plt.legend()

    plt.show()

plot\_train\_history(history,'model results')



!mkdir saved\_model\_result

model.save('saved\_model\_result/final\_model.h5')

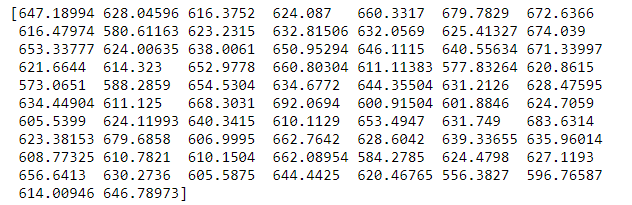
model=load\_model('saved\_model\_result/final\_model.h5')

x\_train=np.reshape(x\_val\_single[0],(1,120,3))

x=model.predict(x\_train)

data=scale.inverse\_transform(x)[0][0:]

print(data)



**User Interfrace:**



Plot Results

Predict Output

Load Model

Validd

Save Model

User Enters data

Validating time series forecasting model

Predicted train data and test data perfomance

Learned time series forecasting model

Train time series forecasting model

Test Data

Train Data

Scaling Features to [0,1]

Data Transformation

**Dataset**