

**TEMPERATURE PREDICTION AND AUTHENTICATING THE CLAIM  
OF VARIOUS SCIENTISTS USING CLIMATE TEMPERATURE**

*An Project report submitted in partial fulfillment of requirements for the award  
of the Degree of*

**BACHELOR OF TECHNOLOGY**

*In*

**COMPUTER SCIENCE AND ENGINEERING**

*By*

PAPPU SRI VINAY TEJA (170030975)

Under the esteemed guidance of

**M. Sai Teja**

Project Coordinator

Skill Geek Pvt. Ltd.



**Department of Computer Science and Engineering**

**KL University**

Green Fields, Vaddeswaram, Tadepalli

Guntur District -522 5022, Andhra Pradesh

**Skill Geek Pvt. Ltd.**

**8-2-293/82/J-111/79 ROAD NO-71, Jubilee Hills**

**Hyderabad**



**Bonafide Certificate**

This is to certify that Practice school project work entitled “Temperature Prediction and authenticating the claim of various scientists using Climate Temperature data” being submitted to department of Computer Science and Engineering, KL University by “Pappu Sri Vinay Teja (170030975)” partial fulfillment of requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide work carried out by them under my supervision.

**Mr. M. Sai Teja,  
Project Coordinator,  
Skill Geek Pvt. Ltd.**

**K L E F (Deemed to be University)**  
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**Certificate**

This is Certified that the project entitled “Temperature Prediction and authenticating the claim of various scientists using Climate Temperature data” which is a PRACTICAL work carried out by Pappu Sri Vinay Teja (170030975) in partial fulfillment for the award of the degree of Bachelor of Technology in Department Computer Science & Engineering, during the year 2020-2021. The project has been approved as it satisfies the academic requirements.

**Department Coordinator**

**(Mr. M. L. Phaneendra)**

**Head of the Department**

**(Dr. V. Hari Kiran)**

**University Guide**

**(Mr. Chand Pasha)**

## Acknowledgement

The satisfaction and euphoria the accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned our efforts our gratitude to all those who helped us in the course of project.

Firstly, I thank the almighty and our parents for providing the moral support needed for carrying out the project work in **Practice School**.

We express our gratitude to our company guide Mr. M. Sai Teja sir (Project Coordinator) for providing us the opportunity to be a part of this project and guiding us throughout patiently.

We are also very thankful to our university guide Mr.Chand Pasha sir for guiding and supporting us during the practice school period.

PAPPU SRI VINAY TEJA

ID NO: 170030975

B. Tech (CSE)

# Table of Contents

I.	Abstract	6
II.	Introduction	7
III.	Analysis	8 - 12
IV.	Implementation Methodology	13
V.	Architecture Pattern Used	14 - 16
VI.	Technologies Employed	17
VII.	Source Code	18 - 27
VIII.	Screenshots	28 - 56
IX.	Conclusion	57
X.	Future Scope	58
XI.	References	59-60

## I. ABSTRACT

Our project is to understanding the “Climate Change Phenomenon” and “authenticating” various claim of Scientist using temperature data. Also “Predictions the temperature”. Temperature is a physical property of a matter it changes rapidly. The main reason behind the temperature change are “Greenhouse Gases”. First, the Greenhouse gases are Carbon Dioxide, Methane, Nitrous Oxide, Fluorinated Gases, ozone. In 18th century the industrial revolution was started from that time the CO<sub>2</sub> increased more than 20% in less than 40 years by industries. In 2018, 6,677 Million Metric Tons of CO<sub>2</sub> equivalent was emitted by factories. Due to more amount of CO<sub>2</sub> more amount of light was reflecting to earth surface. Due to more light more heat is generating which lead to “Global Warming”. So, our aim is to prove that the unnecessary CO<sub>2</sub> released by factory and other human tasks which leads to danger to life on earth. To prove that firstly, we find and correct the errors in the dataset. Fill the missing data with “Mode Method”. After that for analyses and prediction of temperature and causes we are using “Linear Regression, Random Forest, Feedforward Neural Networking and Navies Bayes Classification”. Finally, our outcome is to prove the claim of the scientist and predict the “Temperature and Causes” due to change in temperature.

## **II. INTRODUCTION**

Temperature is a physical property of a matter it changes rapidly. The main reason behind the temperature change are “Greenhouse Gases”. First, the Greenhouse gases are Carbon Dioxide, Methane, Nitrous Oxide, Fluorinated Gases, ozone.

The “Ozone” stops the dangerous light beams release by sun and reflect the remaining light beams to earth surface.

The reflected light beams again reflect back to sky. Then the “CO<sub>2</sub>,N<sub>2</sub>O, H<sub>2</sub>O,CH<sub>4</sub>,CFC-12 ” absorbing and allow the unwanted light to escape from earth atmosphere that required amount of light beams which keep earth surface warm. This process is called as “Greenhouse Effect” discover by “Irish physicist John Tyndall” in 1859.

In 18th century the industrial revolution was started from that time the CO<sub>2</sub> increased more than 20% in less than 40 years by industries. In 2018, 6,677 Million Metric Tons of CO<sub>2</sub> equivalent was emitted by factories.

“Irish physicist John Tyndall and other scientists” said that, due to more amount of CO<sub>2</sub> more number of light beams can escape from earth surface. Due to more light more heat is generating which lead to “Global Warming”.

Again, the Global warming leads to “increase of sea levels, more summer season, wild fire, increase in atmospheric temperature, melting of glaciers, etc. So, CO<sub>2</sub> released by factory which release the CO<sub>2</sub> leads to danger to life on earth.

But, most of the “industrialist and some scientists” says that the Co<sub>2</sub> released by industrials are in limit it not leads to danger to life on earth.

So, our aim is to prove that the unnecessary CO<sub>2</sub> released by factory and other human tasks which release the CO<sub>2</sub> leads to danger to living beings on earth.

### **III. ANALYSIS**

#### **LINEAR REGRESSION:**

Linear Regression can be considered a Machine Learning algorithm that allows us to map numeric inputs to numeric outputs, fitting a line into the data points. Linear Regression is one of the most famous topics in both statistics and Machine Learning. It is so essential to the point where it withhold a significant part in almost any Machine Learning course out there. Linear Regression create a graph with the input and find the line that best fits the data points on the plot.

Straight line on a plot follows the formula:  $M \cdot X + B$ . Where  $M$  is the slope of the line,  $B$  is the y-intercept that allows vertical movement of the line, and  $X$  which is the function's input value.

#### **Real Life Application:**

1. Businesses often use linear regression to understand the relationship between advertising spending and revenue.
2. Medical researchers often use linear regression to understand the relationship between drug dosage and blood pressure of patients.
3. Agricultural scientists often use linear regression to measure the effect of fertilizer and water on crop yields.
4. Data scientists for professional sports teams often use linear regression to measure the effect that different training regimens have on player performance.

#### **Advantages of Linear Regression:**

1. Linear Regression is simple to implement and easier to interpret the output coefficients.
2. When you know the relationship between the independent and dependent variable have a linear relationship, this algorithm is the best to use because of its less complexity compared to other algorithms.

### **Disadvantages of Linear Regression:**

1. On the other hand in linear regression technique outliers can have huge effects on the regression and boundaries are linear in this technique.
2. Diversely, linear regression assumes a linear relationship between dependent and independent variables. That means it assumes that there is a straight-line relationship between them. It assumes independence between attributes.

### **RANDOM FOREST:**

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean/average prediction of the individual trees. Random forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because of its simplicity and diversity.

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

### **Real Life Application:**

1. In the healthcare domain it is used to identify the correct combination of components in medicine and to analyze a patient's medical history to identify diseases.
2. Random forest is used in e-commerce to determine whether a customer will actually like the product or not.

### **Advantages of Linear Regression:**

1. It reduces overfitting in decision trees and helps to improve the accuracy.
2. It is flexible to both classification and regression problems.
3. It works well with both categorical and continuous values.

### **Disadvantages of Linear Regression:**

1. It requires much computational power as well as resources as it builds numerous trees to combine their outputs.
2. It also requires much time for training as it combines a lot of decision trees to determine the class.
3. Due to the ensemble of decision trees, it also suffers interpretability and fails to determine the significance of each variable.

### **FEED FORWARD NEURAL NETWORKS:**

A feedforward neural network is a biologically inspired classification algorithm. It consists of a number of simple neuron-like processing units, organized in layers. Every unit in a layer is connected with all the units in the previous layer.

The structure of a neural-network algorithm has three layers:

1. The first layer is input layer. It feeds past data values into the next (hidden) layer.
2. The second layer is hidden layer. It encapsulates several complex functions that create predictors; often those functions are hidden from the user. The hidden layer represents mathematical functions that modify the input data these functions are called neurons.
3. The final layer is output layer collects the predictions made in the hidden layer and produces the final result.

### **Real Life Application:**

1. Image Processing and Character recognition
2. Forecasting

### **Advantages of Feed Forward Neural Networks:**

1. ANNs have the ability to learn and model non-linear and complex relationships.
2. ANNs can generalize. After learning from the initial inputs and their relationships, it can infer unseen relationships on unseen data as well, thus making the model generalize and predict on unseen data.

### **Disadvantages of Feed Forward Neural Networks:**

1. Neural networks have their “black box” nature. Simply put, you don’t know how or why your NN came up with a certain output.
2. Neural networks usually require much more data than traditional machine learning algorithms, as in at least thousands if not millions of labeled samples. This isn’t an easy problem to deal with and many machine learning problems can be solved well with less data if you use other algorithms.

### **DECISION TREE:**

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

There are three stages in decision tree:

1. **Splitting:** The process of partitioning the data set into subsets. Splits are formed on a particular variable.
2. **Pruning:** The shortening of branches of the tree. Pruning is the process of reducing the size of the tree by turning some branch nodes into leaf nodes, and removing the leaf nodes under the original branch.
3. **Tree Selection:** The process of finding the smallest tree that fits the data. Usually this is the tree that yields the lowest cross-validated error.

### **Real Life Application:**

1. Decision trees involve evaluating prospective growth opportunities for businesses based on historical data.
2. Decision trees to predict the probability of a customer defaulting on a loan, by applying predictive model generation using the client’s past data.

### **Advantages of Decision Tree:**

1. Decision trees is that their outputs are easy to read and interpret, without even requiring statistical knowledge.
2. Compared to other decision techniques, decision trees take less effort for data preparation.

### **Disadvantages of Decision Tree:**

1. One of the limitations of decision trees is that they are largely unstable compared to other decision predictors.
2. In addition, decision trees are less effective in making predictions when the main goal is to predict the outcome of a continuous variable.

## **IV. IMPLEMENTATION METHODOLOGY**

### **IDE:**

Python IDLE, Kaggle Notebook(python)

### **API's:**

Kaggle

### **Dataset:**

Climate Change: Earth Surface Temperature Data from Kaggle

### **Algorithms:**

Linear Regression, Random Forest Regression, Feedforward Neural Networking, Decision Tree

### **Technologies:**

Python

## **V. ARCHITECTURE PATTERN USED**

Architecture Pattern Used in this project is “Software development Life Cycle”.

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

### **Stage 1: Planning and Requirement Analysis**

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

## **Stage 2: Defining Requirements**

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

## **Stage 3: Designing the Product Architecture**

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

## **Stage 4: Building or Developing the Product**

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code.

Different high-level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

### **Stage 5: Testing the Product**

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

### **Stage 6: Deployment in the Market and Maintenance**

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

## VI. TECHNOLOGIES EMPLOYED

### **Python IDLE:**

IDLE is an integrated development environment for Python, which has been bundled with the default implementation of the language since 1.5.2b1. It is packaged as an optional part of the Python packaging with many Linux distributions. It is completely written in Python and the Tkinter GUI toolkit. IDLE is intended to be a simple IDE and suitable for beginners, especially in an educational environment. To that end, it is cross-platform, and avoids feature clutter.

### **Kaggle Notebook (python):**

Kaggle, a subsidiary of Google LLC, is an online community of data scientist and machine practitioners. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

## VII. SOURCE CODE

### IMPORTED PACKAGE

```
import math  
import numpy as np  
import pandas as pd  
from pandas import DataFrame  
import matplotlib.pyplot as plt  
import sklearn  
from sklearn.model_selection import train_test_split  
from sklearn.metrics import accuracy_score  
from sklearn.metrics import mean_absolute_error  
from sklearn.linear_model import LinearRegression  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.tree import DecisionTreeRegressor  
from sklearn import preprocessing  
from numpy import random
```

### LOADING THE INPUT INTO DATA

```
data = pd.read_csv('C:/Users/psriv/OneDrive/Documents/PRACTICE SCHOOL/Done  
Doc/Project/GlobalTemperatures.csv');  
print(data)
```

## **FINDING THE MISSING VALUES**

```
missing_data = data.isnull()  
print(missing_data)
```

## **SUM OF MISSING VALUES**

```
missing_data_sum = data.isnull().sum()  
print(missing_data_sum)
```

## **SEPERATING THE YEAR FROM DATE COLUMN**

```
data['year'] = pd.DatetimeIndex(data['dt']).year
```

## **STARTING OF LAND AND OCEAN TEMPERATURE**

```
sep_loat_data=data[['year','LandAndOceanAverageTemperature','LandAndOceanAverage  
TemperatureUncertainty']]  
  
print(sep_loat_data)  
sep_loat_data.columns
```

## **STORING THE REQUIRED DATA INTO NEW DATA**

```
sep_loat_data.to_csv('LandAndOceanAverageTemperature Data')
```

## **PRINTING THE LOADED DATA**

```
loat_data=pd.read_csv('LandAndOceanAverageTemperature Data')  
print(loat_data)
```

## **COUNT OF ELEMENTS IN LAND OCEAN AVERAGE TEMPERATURE**

```
LandAndOceanAverageTemperature_count=loat_data['LandAndOceanAverageTemperature'].value_counts()  
print(LandAndOceanAverageTemperature_count)
```

## **MAX VALUE IN THE COUNT OF ELEMENTS IN LAND OCEAN AVERAGE TEMPERATURE**

```
LandAndOceanAverageTemperature_count_max=  
loat_data['LandAndOceanAverageTemperature'].value_counts().idxmax()  
print(LandAndOceanAverageTemperature_count_max)
```

## **FILL THE NULL IN LAND OCEAN AVERAGE TEMPERATURE**

```
print('FILL THE NULL VALUES OF LAND AND OCEAN AVERAGE  
TEMPERATURE WITH THE MAX VALUE OF THE COUNT OF ELEMENTS IN  
LAND AND OCEAN AVERAGE TEMPERATURE')
```

```
LandAndOceanAverageTemperature_fillnull =  
loat_data['LandAndOceanAverageTemperature'].fillna(LandAndOceanAverageTemperature_count_max,inplace=True)  
print(loat_data['LandAndOceanAverageTemperature'])
```

## **THE COUNT OF ELEMENTS IN LAND OCEAN AVERAGE TEMPERATURE UNCERTANITY**

```
LandAndOceanAverageTemperatureUncertainty_count=loat_data['LandAndOceanAverageTemperatureUncertainty'].value_counts()  
print(LandAndOceanAverageTemperatureUncertainty_count)
```

## **MAX VALUE IN THE COUNT OF ELEMENTS IN AVERAGE TEMPERATURE UNCERTANINTY**

```
LandAndOceanAverageTemperatureUncertainty_count_max=
loat_data['LandAndOceanAverageTemperatureUncertainty'].value_counts().idxmax()
print(LandAndOceanAverageTemperatureUncertainty_count_max)
```

## **FILL THE NULL IN LAND OCEAN AVERAGE TEMPERATURE**

```
print('FILL THE NULL VALUES OF LAND AND OCEAN AVERAGE TEMPERATURE UNCERTAINTY WITH THE MIN VALUE OF THE COUNT OF ELEMENTS IN LAND AND OCEAN AVERAGE TEMPERATURE')
```

```
LandAndOceanAverageTemperatureUncertainty_fillnull=loat_data['LandAndOceanAverageTemperatureUncertainty'].fillna(LandAndOceanAverageTemperatureUncertainty_count_max,inplace=True)
```

```
print(loat_data['LandAndOceanAverageTemperatureUncertainty'])
```

## **PLOTTING A GRAPH BETWEEN YEAR AND AVERAGE TEMPERATURE**

```
x=loat_data['year']
y=loat_data['LandAndOceanAverageTemperature']
plt.scatter(x,y, color='red')
plt.title('Year Vs LandAndOceanAverageTemperature', fontsize=14)
plt.xlabel('year', fontsize=14)
plt.ylabel('LandAndOceanAverageTemperature', fontsize=14)
plt.grid(True)
plt.show()
```

## **PLOTTING A BAR GRAPH BETWEEN YEAR AND AVERAGE TEMPERATURE**

```
x=loat_data['year']
y=loat_data['LandAndOceanAverageTemperature']
fig = plt.figure(figsize =(10, 7))
plt.bar(x,y)
# Show Plot
plt.show()
```

## **CONVERTING THE VALUES FLOAT TO INT OF LAND AND OCEAN TEMPERATURE**

```
loat_int=loat_data['LandAndOceanAverageTemperature'].astype(int)
print(loat_int)
```

## **FINDING THE MOST COMMONLY REPEATED VALUE OF LAND AND OCEAN TEMPERATURE**

```
loat_int_counts=loat_int.value_counts()
print(loat_int_counts)
```

## **PLOTTING BAR GRAPH OF LINEAR REGRESSION LAND AVERAGE TEMPERATURE**

```
loat_int_counts.plot.barh(figsize=(25,10))
```

## **X AND y**

```
temp_loat_data=loat_data[list(loat_data.dtypes[loat_data.dtypes!='objects'].index)]  
X=temp_loat_data  
y=temp_loat_data.pop('LandAndOceanAverageTemperature')
```

## **TRAINING AND TESTING X AND y**

```
train_X,test_X,train_y,test_y=train_test_split(X,y,test_size=0.2,random_state=4)  
print('train_X.head')  
print(train_X.head())  
print('train_y.head')  
print(train_y.head())
```

## **LINEAR REGRESSION**

```
lr=LinearRegression()  
lr.fit(train_X,train_y)  
loat_lr_prediction = lr.predict(test_X)  
loat_lr_Final=pd.DataFrame({'actual':test_y,'prediction' : loat_lr_prediction,'error':(test_y-  
loat_lr_prediction)})  
print('Final Linear regression predicted values')  
print(loat_lr_Final)
```

## **PLOTTING BAR GRAPH OF LINEAR REGRESSION LAND OCEAN AVERAGE TEMPERATURE**

```
loat_lr_Final.plot.bar(figsize=(25,10))
```

## **CONVERTING THE VALUES FLOAT TO INT OF LINEAR REGRESSION LAND OCEAN AVERAGE TEMPERATURE**

```
loat_lr_Final_int=loat_lr_Final.astype(int)  
print(loat_lr_Final_int)
```

## **FINDING THE MOST COMMONLY REPEATED VALUE OF LINEAR REGRESSION LAND OCEAN AVERAGE TEMPERATURE**

```
loat_lr_Final_prediction_value_counts=loat_lr_Final_int['prediction'].value_counts()
```

## **PLOTTING BAR GRAPH OF LINEAR REGRESSION LAND OCEAN AVERAGE TEMPERATURE**

```
loat_lr_Final_prediction_value_counts.plot.bar(figsize=(25,10))
```

## **RANDOM FOREST REGRESSION**

```
rf_model= RandomForestRegressor()  
# fit your model  
rf_model.fit(train_X, train_y)
```

## **CALCULATE THE MEAN ABSOLUTE ERROR OF YOUR RANDOM FOREST MODEL ON THE VALIDATION DATA**

```
loat_rf_prediction = rf_model.predict(test_X)  
loat_rf_Final=pd.DataFrame({'actual':test_y,'prediction' :  
    loat_rf_prediction,'erroe':(test_y-loat_rf_prediction)})  
print(loat_rf_Final)
```

## **PLOTTING BAR GRAPH OF RANDOM FOREST REGRESSION**

### **LAND OCEAN AVERAGE TEMPERATURE**

```
loat_rf_Final.plot.bar(figsize=(25,10))
```

## **CONVERTING THE VALUES FLOAT TO INT OF RANDOM FOREST REGRESSION LAND OCEAN AVERAGE TEMPERATURE**

```
loat_rf_Final_int=loat_rf_Final.astype(int)
```

```
print(loat_rf_Final_int)
```

## **FINDING THE MOST COMMONLY REPEATED VALUE OF LINEAR REGRESSION LAND OCEAN AVERAGE TEMPERATURE**

```
loat_rf_Final_prediction_value_counts = loat_rf_Final_int['prediction'].value_counts()
```

```
loat_lr_Final_prediction_value_counts.plot.bah(figsize=(25,10))
```

## **DECISION TREE**

```
dt = DecisionTreeRegressor(max_leaf_nodes=100, random_state=1)
```

```
# fit your model
```

```
dt.fit(train_X, train_y)
```

```
loat_dt_prediction = dt.predict(test_X)
```

```
loat_dt_Final=pd.DataFrame({'actual':test_y,'prediction' : loat_dt_prediction,'diff':(test_y-loat_dt_prediction)})
```

```
print(loat_dt_Final)
```

## **PLOTTING BAR GRAPH OF RANDOM FOREST REGRESSION**

### **LAND OCEAN AVERAGE TEMPERATURE**

```
loat_dt_Final.plot.bar(figsize=(25,10))
```

## **CONVERTING THE VALUES FLOAT TO INT OF RANDOM FOREST REGRESSION LAND OCEAN AVERAGE TEMPERATURE**

```
loat_dt_Final_int=loat_dt_Final.astype(int)  
print(loat_dt_Final_int)
```

## **FINDING THE MOST COMMONLY REPEATED VALUE OF LINEAR REGRESSION LAND OCEAN AVERAGE TEMPERATURE**

```
loat_dt_Final_prediction_value_counts = loat_dt_Final_int['prediction'].value_counts()  
loat_dt_Final_prediction_value_counts.plot.barh(figsize=(25,10))
```

## **FEEDFORWARD NEURAL NETWORK**

### **CONVERT DATASET INTO ARRAY**

```
x=loat_data['LandAndOceanAverageTemperature'].to_numpy()  
print(x)  
s_ip = x.size  
print(s_ip)  
Y = []  
for i in range(s_ip):  
    Y.append(random.randint(-20, 20))  
print(Y)  
b1,b2,b3=0.1,0.1,0.1  
#initializing the input weights  
w1=0.2  
w2=0.2  
w3=0.2
```

```

w4=0.2
#initializing th hidden layer weights
w5=0.2
w6=0.2
for i in range(0, len(x)):
    #calculating net input for hidden layer z1
    z1=x[i]*w1+x[i]*w3+b1
    #calculating net input for hidden layer z2
    z2=x[i]*w2+x[i]*w4+b2
    outz1=1/(1+math.exp(-z1))
    outz2=1/(1+math.exp(-z2))
    t_out=outz1*w5+outz2*w6+b3
    outy=1/(1+math.exp(-t_out))
    total_error=(1/2)*(Y[i]-outy)**2
    print('the output of z1 in the hidden layer is',outz1)
    print('the output of z2 in the hidden layer is',outz2)
    print('net output of y is',outy)
    print('the total error is',total_error)
    print('net output of y is',outy)

```

## **END OF LAND AVERAGE TEMPERATURE AND LAND OCEAN AVERAGE TEMPERATURE**

## VIII. SCREEN SHOTS

A screenshot of a Microsoft Edge browser window displaying a Kaggle notebook titled "Global". The notebook interface shows a sidebar with navigation links like Home, Compete, Data, Notebooks, Discuss, Courses, and More. Below the sidebar, a "Recently Viewed" section lists "Global", "Climate Change: Earth ...", "Country", "State", and "Major City". The main content area displays a Jupyter Notebook cell containing a table of temperature data. The table has columns: dt, LandAverageTemperature, LandAverageTemperatureUncertainty, LandMaxTemperature, LandMaxTemperatureUncertainty, LandMinTemperature, LandMinTemperatureUncertainty, and LandAndOceanAverageTemperature. The data shows values for various dates from 1750-01-01 to 2015-12-01, including some NaN and null values. A vertical sidebar on the right shows "Version 7 of 7" and "Notebook" sections with tabs for Input (1), Execution Info, Log, and Comments (0). The URL in the address bar is <https://www.kaggle.com/pappusrinivayteja/global>.

Fig 0.1: Loading the data

A screenshot of a Microsoft Edge browser window displaying a Kaggle notebook titled "Global". The notebook interface shows a sidebar with navigation links like Home, Compete, Data, Notebooks, Discuss, Courses, and More. Below the sidebar, a "Recently Viewed" section lists "Global", "Climate Change: Earth ...", "Country", "State", and "Major City". The main content area displays a Jupyter Notebook cell containing a table of boolean data. The table has columns: dt, LandAverageTemperature, LandAverageTemperatureUncertainty, LandMaxTemperature, LandMaxTemperatureUncertainty, LandMinTemperature, LandMinTemperatureUncertainty, and LandAndOceanAverageTemperature. The data shows values for various dates from 1750-01-01 to 2015-12-01, mostly False values with some True values interspersed. A vertical sidebar on the right shows "Version 7 of 7" and "Notebook" sections with tabs for Input (1), Execution Info, Log, and Comments (0). The URL in the address bar is <https://www.kaggle.com/pappusrinivayteja/global>.

Fig 0.2: Finding the null values

The screenshot shows a Microsoft Edge browser window displaying a Kaggle notebook titled "Global". The notebook is using data from "Climate Change: Earth Surface Temperature Data". The code in In [4] is:

```
#SUM OF MISSING VALUES
#
#
missing_data_sum = data.isnull().sum()
print(missing_data_sum)
```

The output of this code is:

	dt	LandAverageTemperature	LandAverageTemperatureUncertainty	LandMaxTemperature	LandMaxTemperatureUncertainty	LandMinTemperature	LandMinTemperatureUncertainty	LandAndOceanAverageTemperature	LandAndOceanAverageTemperatureUncertainty
0	0	12	12	1200	1200	1200	1200	1200	1200

The code in In [5] is:

```
#SEPARATING THE YEAR FROM DATE COLUMN
data['year'] = pd.DatetimeIndex(data['dt']).year
```

Fig 0.3: Count of null values

The screenshot shows a Microsoft Edge browser window displaying a Kaggle notebook titled "Global". The notebook is using data from "Climate Change: Earth Surface Temperature Data". The code in In [7] is:

```
sep_lat_data = data[['year', 'LandAverageTemperature', 'LandAverageTemperatureUncertainty']]
print(sep_lat_data)
```

The output of this code is:

	year	LandAverageTemperature	LandAverageTemperatureUncertainty
0	1750	3.934	3.574
1	1750	3.083	3.702
2	1750	5.626	3.076
3	1750	8.490	2.451
4	1750	11.573	2.072
...	...	...	...
3187	2015	14.755	0.072
3188	2015	12.999	0.079
3189	2015	10.801	0.102
3190	2015	7.433	0.119
3191	2015	5.518	0.100

The code in In [8] is:

```
sep_lat_data.columns
```

Fig 0.4: Separating the data

The screenshot shows a Microsoft Edge browser window displaying a Kaggle notebook titled "Global". The notebook interface includes a sidebar with navigation links like Home, Compete, Data, Notebooks, Discuss, Courses, and More. The main area shows a code cell (In [10]) with the following Python code:

```
#PRINTING THE LOADED DATA
lat_data = pd.read_csv('LandAvgTemp_Data')
print(lat_data)
```

The output of the code is displayed in two parts. The first part shows the first few rows of the "lat\_data" DataFrame:

	Unnamed: 0	year	LandAverageTemperature
0	0	1750	3.034
1	1	1750	3.083
2	2	1750	5.626
3	3	1750	8.490
4	4	1750	11.573
...	...	...	...
3187	3187	2015	14.755
3188	3188	2015	12.999
3189	3189	2015	10.801
3190	3190	2015	7.433
3191	3191	2015	5.518

The second part shows the "LandAverageTemperatureUncertainty" column:

	LandAverageTemperatureUncertainty
0	3.574
1	3.702
2	3.076
3	2.451
4	2.072
...	...
3187	0.072
3188	0.079
3191	0.102

On the right side of the notebook, there is a sidebar with options for "Version 7 of 7", "Notebook", "Input (1)", "Execution Info", "Log", and "Comments (0)".

Fig 0.5: Storing the separated data into new csv file

The screenshot shows a Microsoft Edge browser window displaying a Kaggle notebook titled "Global". The notebook interface includes a sidebar with navigation links like Home, Compete, Data, Notebooks, Discuss, Courses, and More. The main area shows a code cell (In [10]) with the following Python code:

```
#COUNT OF ELEMENTS IN LAND AVERAGE TEMPERATURE
#
#LandAverageTemperature_count = lat_data['LandAverageTemperature'].value_counts()
print(LandAverageTemperature_count)
```

The output of the code is displayed in a table:

LandAverageTemperature	count
13.293	4
13.765	4
13.827	3
12.247	3
13.953	3
..	
3.663	1
13.431	1
3.477	1
0.715	1
14.750	1

Below this, another code cell (In [12]) shows the following code:

```
#MAX VALUE IN THE COUNT OF ELEMENTS IN LAND AVERAGE TEMPERATURE
#
#LandAverageTemperature_count_max = lat_data['LandAverageTemperature'].value_counts().idxma
```

Fig 0.6: Count of the average temperature from new csv file

Global | Kaggle

Would you like to set Microsoft Edge as your default browser? Set as default

**kaggle**

- Home
- Compete
- Data
- Notebooks
- Discuss
- Courses
- More

Recently Viewed

- Global
- Climate Change: Earth ...
- Country
- State
- Major City

View Active Events

Global

Python notebook using data from Climate Change: Earth Surface Temperature Data · 54 views · 1mo ago · Edit tags

ELEMENTS IN LAND AVERAGE TEMPERATURE'

```
LandAverageTemperature_fillnull = lat_data['LandAverageTemperature'].fillna(lat_data['LandAverageTemperature'].count_max,inplace=True)

print(lat_data['LandAverageTemperature'])
```

FILL THE NULL VALUES OF LAND AVERAGE TEMPERATURE WITH THE MAX VALUE OF THE COUNT OF ELEMENTS IN LAND AVERAGE TEMPERATURE

	LandAverageTemperature
0	3.034
1	3.083
2	5.626
3	8.498
4	11.573
...	
3187	14.755
3188	12.999
3189	10.801
3190	7.433
3191	5.518

```
Name: LandAverageTemperature, Length: 3192, dtype: float64
```

In [14]: #THE COUNT OF ELEMENTS IN LAND AVERAGE TEMPERATURE UNCERTAINTY

Fig 0.7: Fill the null values with the highest Count of the average temperature

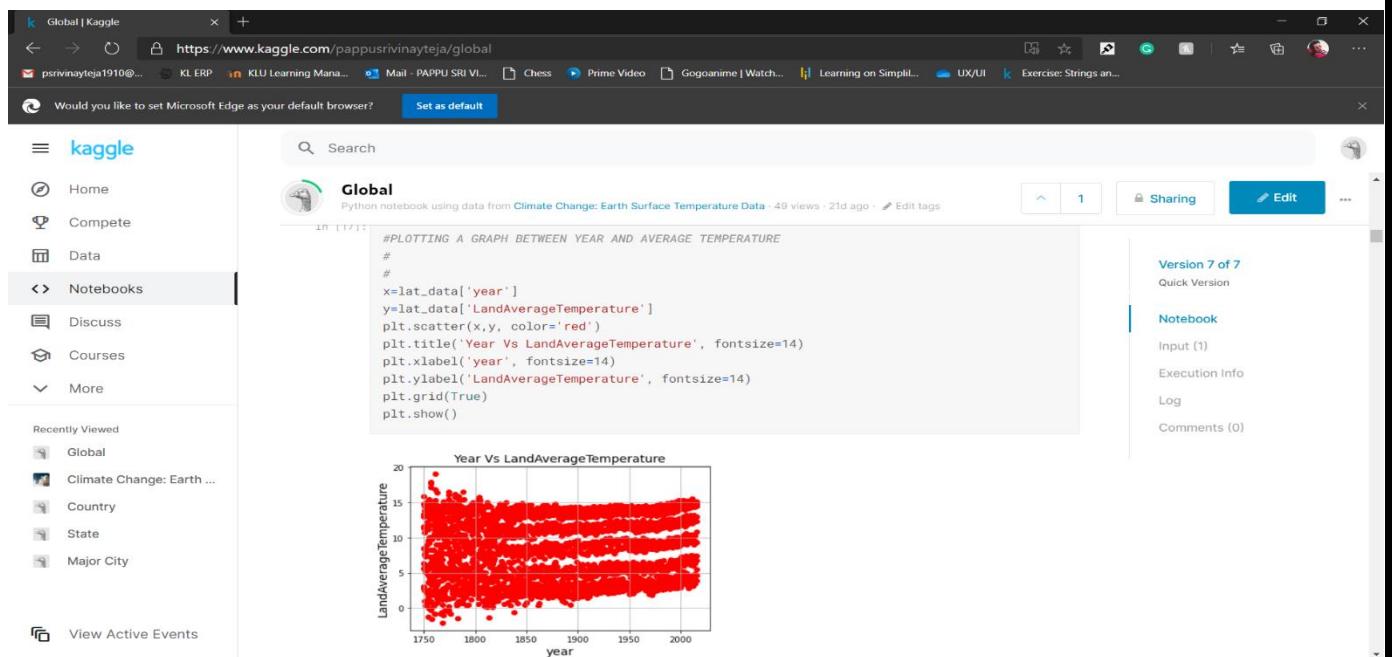


Fig 1.1: Graph of Land Average Temperature (Global Input)

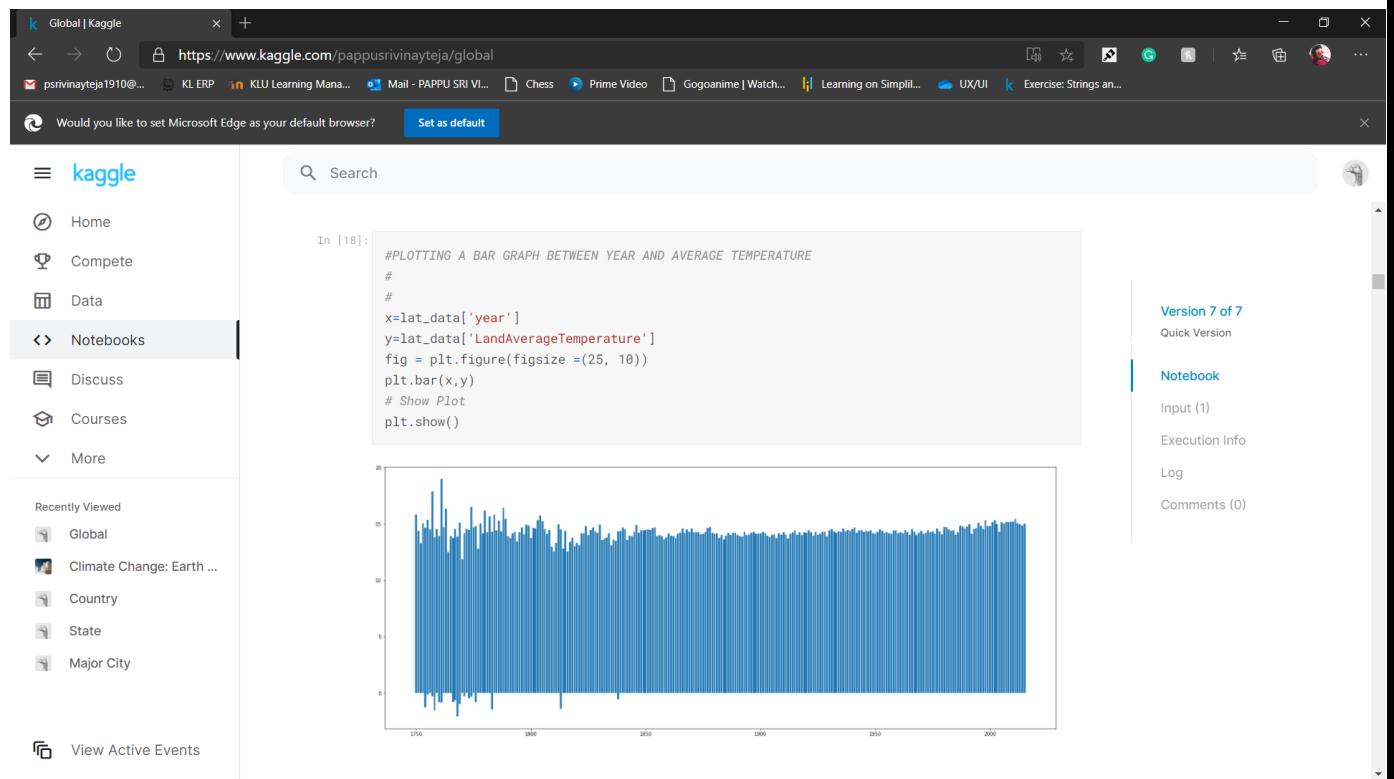


Fig 1.2: Bar Graph of Land Average Temperature (Global Input)

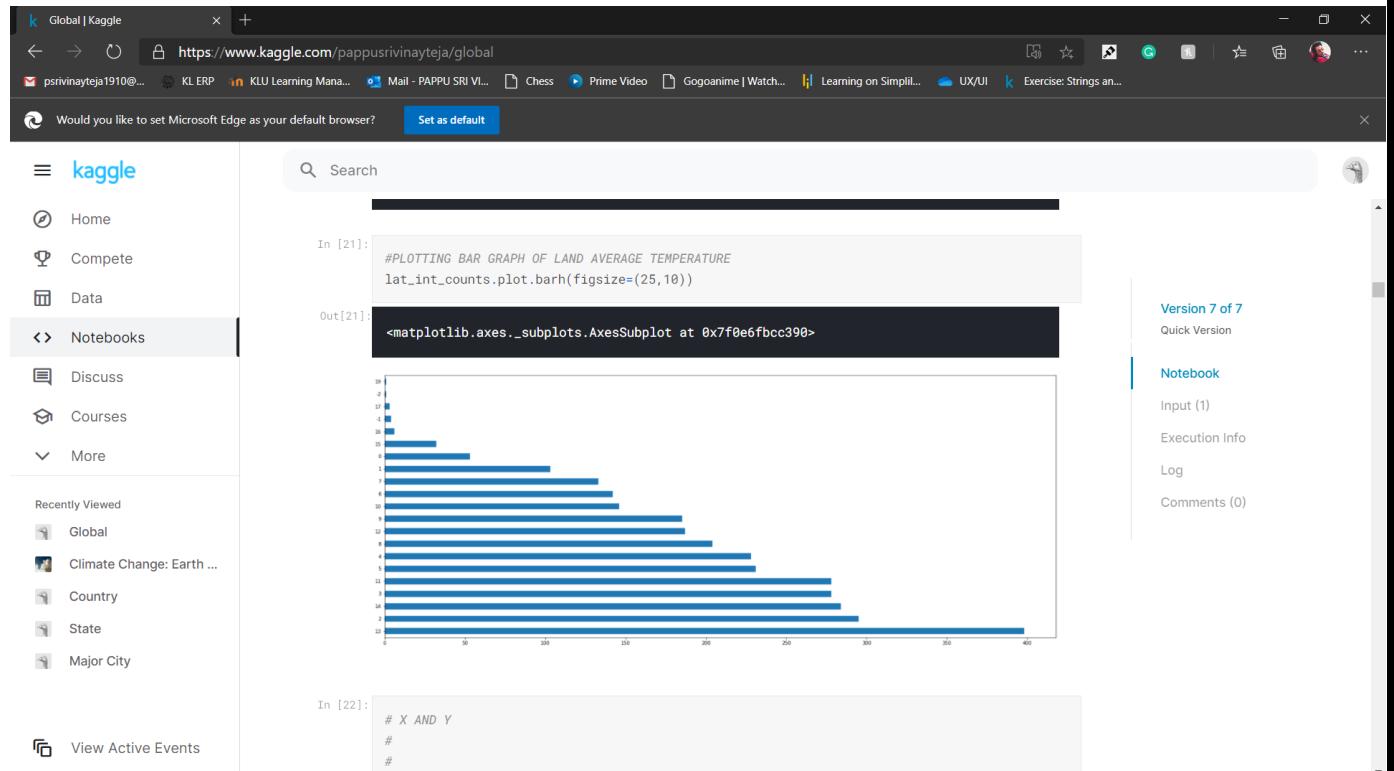


Fig 1.3: Graph of highest count of Land Average Temperature (Global Input)

The screenshot shows a Microsoft Edge browser window displaying a Kaggle notebook titled "Global". The notebook URL is <https://www.kaggle.com/pappusrinayteja/global>. The notebook content displays Python code and its output. The code prints the first few rows of the datasets "train\_X" and "train\_y". The "train\_X.head" output shows columns: Unnamed: 0, year, LandAverageTemperatureUncertainty. The "train\_y.head" output shows the column name "LandAverageTemperature".

```

train_X,test_X,train_y,test_y=train_test_split(X,y,test_size=0.2,random_state=4)
print('train_X.head')
print(train_X.head())
print('\n')
print('train_y.head')
print(train_y.head())

train_X.head
      Unnamed: 0   year  LandAverageTemperatureUncertainty
1804        1804  1900                 0.357
3162        3162  2013                 0.126
79           79   1756                2.175
1728        1728  1894                 0.372
1133        1133  1844                 0.724

train_y.head
      Unnamed: 0
1804    11.385
3162   15.003
79     13.437
1728    1.691
1133   12.849
Name: LandAverageTemperature, dtype: float64

```

Fig 1.4: Training and Test data of Land Average Temperature (Global Input)

The screenshot shows a Microsoft Edge browser window displaying a Kaggle notebook titled "Global". The notebook URL is <https://www.kaggle.com/pappusrinayteja/global>. The notebook content displays Python code for linear regression. The code fits a LinearRegression model to the training data and then prints the predicted values for the test data. The output shows the predicted values along with the actual values and error.

```

#
lr=LinearRegression()
lr.fit(train_X,train_y)

lat_lr_prediction = lr.predict(test_X)

lat_lr_Final=pd.DataFrame({'actual':test_y,'prediction' : lat_lr_prediction,'error':(test_y-lat_lr_prediction)})
print('Final Linear regression predicted values')
print(lat_lr_Final)

Final Linear regression predicted values
      actual  prediction   error
2414  5.091    7.725800 -2.634800
333   6.956    6.822242  0.133758
1471  13.821   9.593263  4.227737
672   2.917    5.985474 -3.068474
3113  14.281   7.880533  6.320467
...
1550   4.924    8.181501 -3.257501
2348   2.525    7.193876 -4.668876
459    8.714    6.285552  2.428448
889    2.679    6.606814 -3.927814
2795   4.327    9.887924 -5.480924

[639 rows x 3 columns]

```

Fig 2.1: Linear Regression of Land Average Temperature (Global Input)

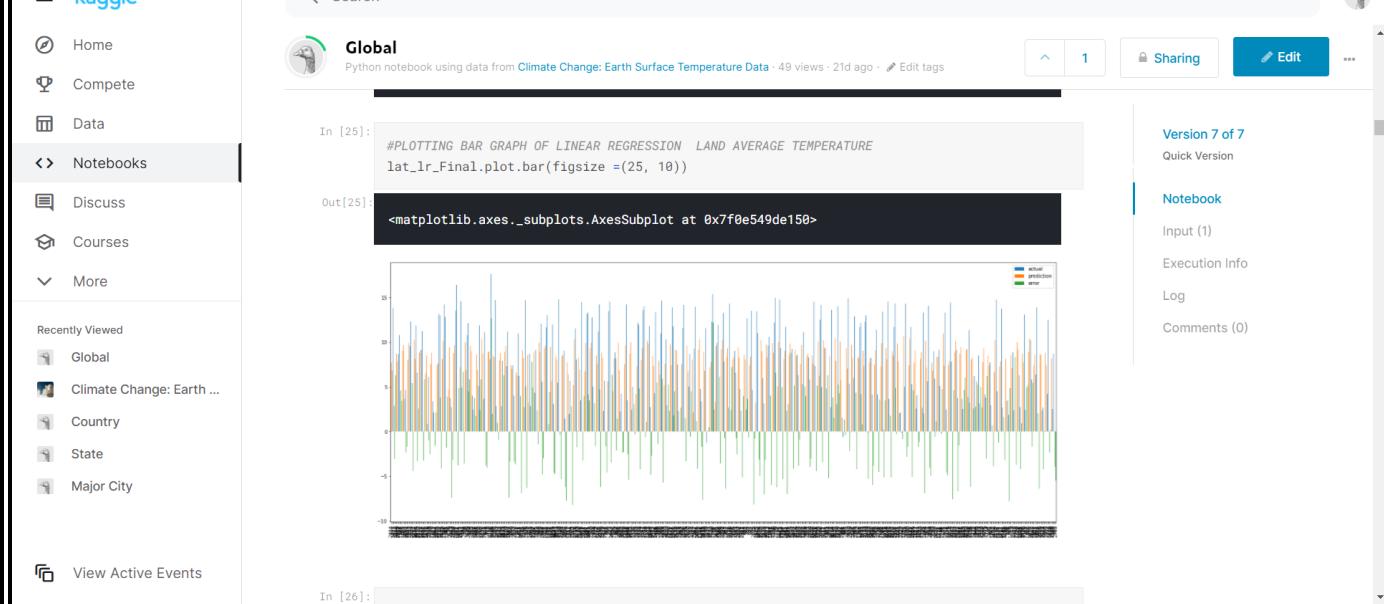


Fig 2.2: Bar Graph of Linear Regression's Land Average Temperature (Global Input)

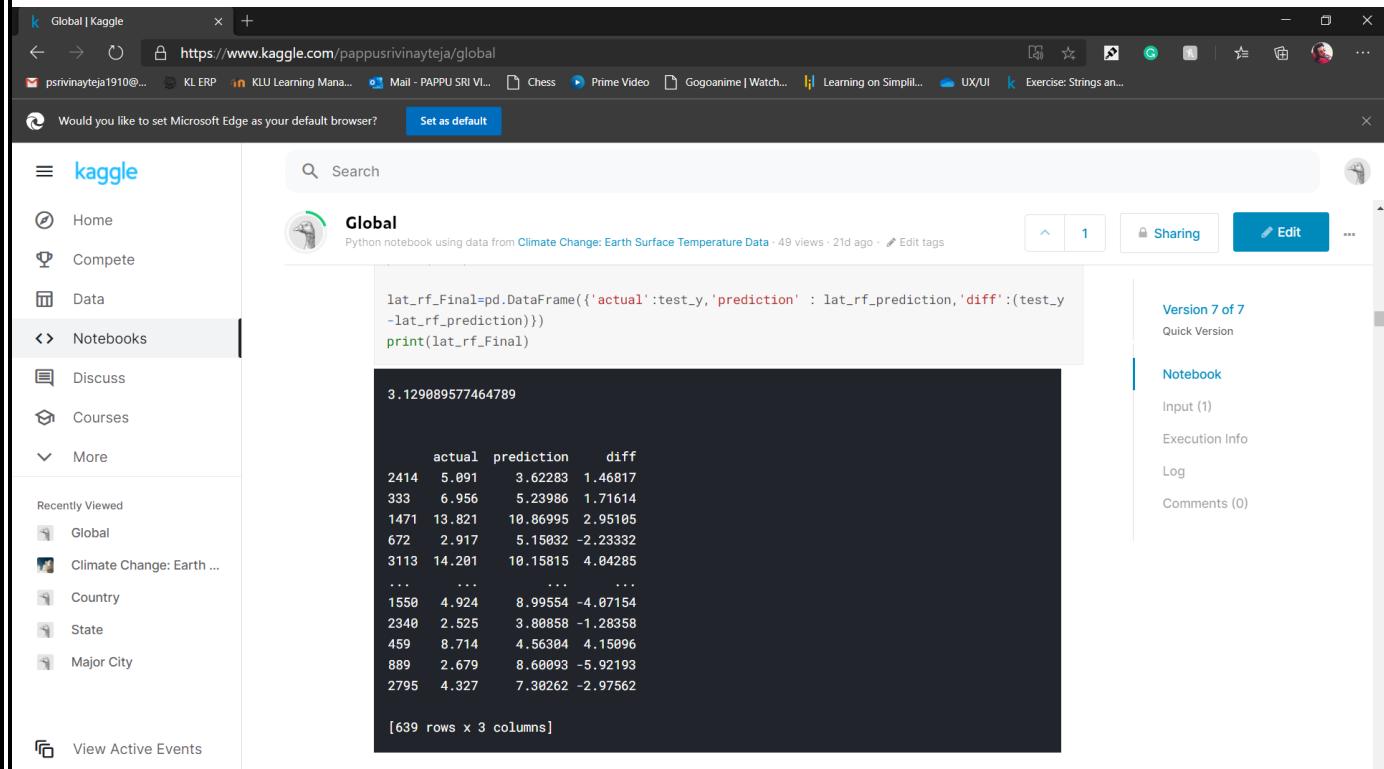


Fig 2.3: Random Forest Regression of Land Average Temperature (Global Input)

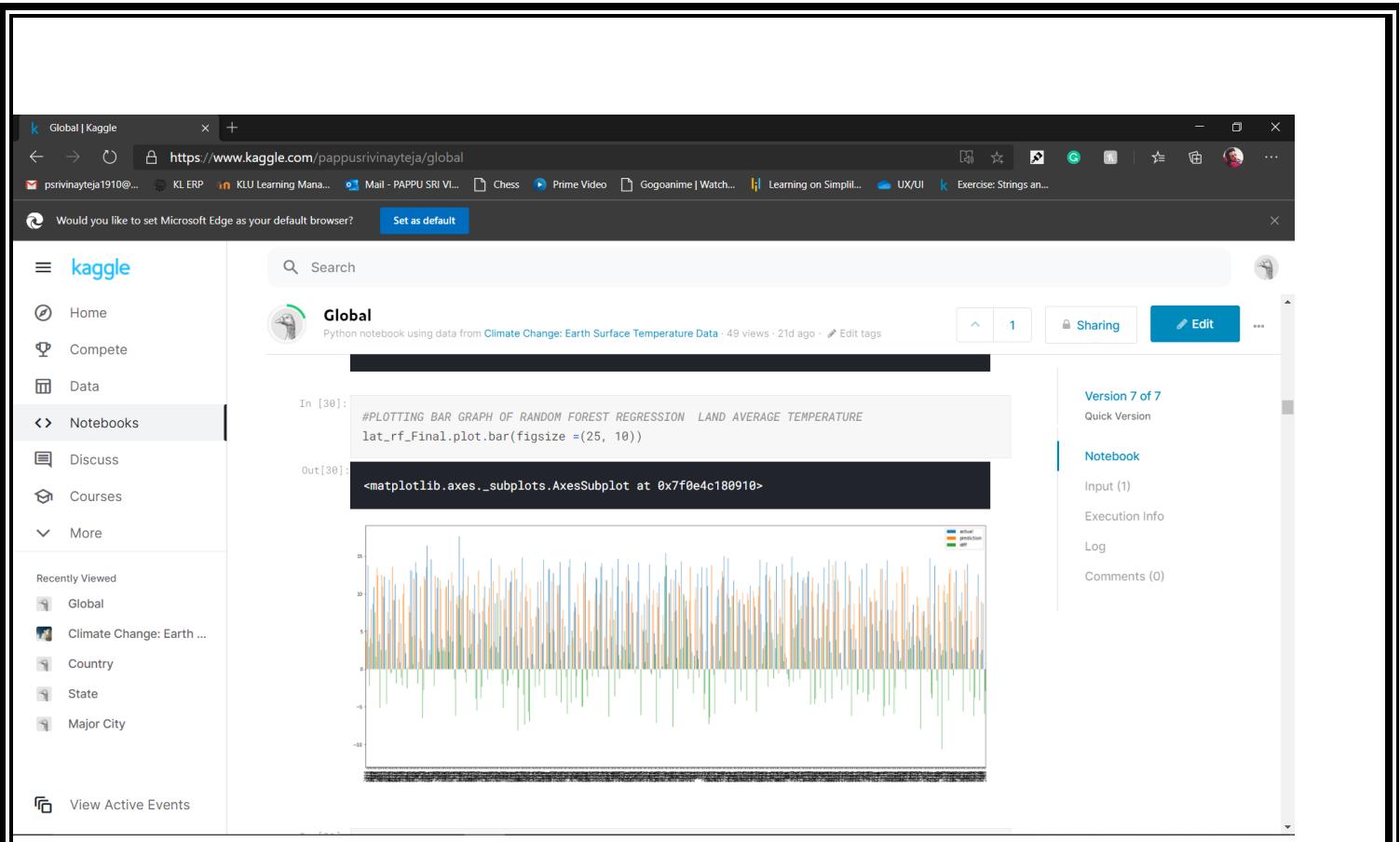


Fig 2.4: Bar Graph of Linear Regression's Land Average Temperature (Global Input)

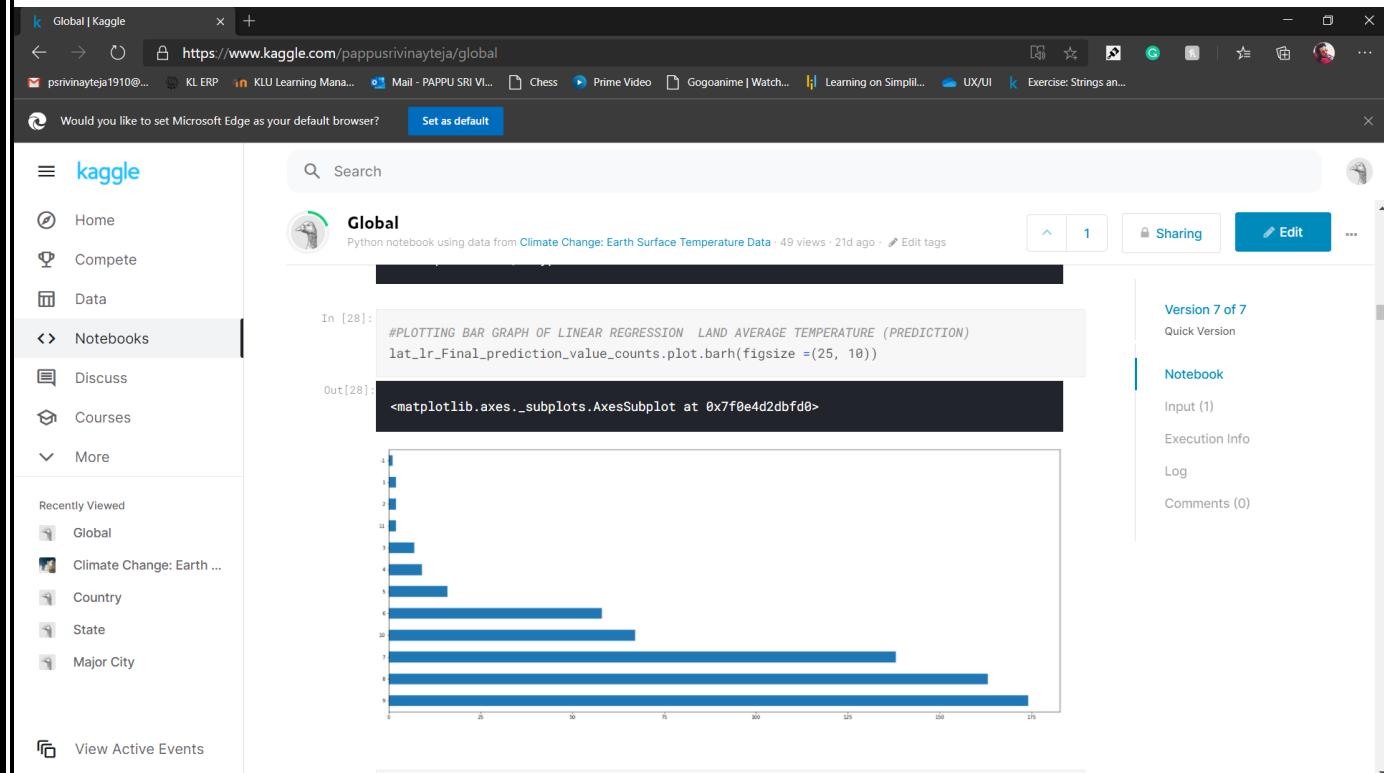


Fig 2.5: Bar Graph of highest count of Random Forest - Land Average Temperature (Global Temperature)

Global | Kaggle

https://www.kaggle.com/pappusrinayteja/global

Would you like to set Microsoft Edge as your default browser? Set as default

**kaggle**

Search

DECISION TREE

```
dt = DecisionTreeRegressor(max_leaf_nodes=100, random_state=1)

# fit your model
dt.fit(train_X, train_y)

lat_dt_prediction = dt.predict(test_X)

lat_dt_Final=pd.DataFrame({'actual':test_y,'prediction' : lat_dt_prediction,'error':(test_y-lat_dt_prediction)})
print(lat_dt_Final)
```

	actual	prediction	error
2414	5.091	5.079789	0.011211
333	6.956	7.417000	-0.461000
1471	13.821	7.664889	6.156111
672	2.917	5.615656	-2.698656
3113	14.281	9.171125	5.029875
...	...	...	...
1550	4.924	11.697333	-6.773333
2340	2.525	5.394895	-2.869895
459	8.714	5.643000	3.071000
889	2.679	5.615656	-2.936656
2795	4.327	8.756187	-4.429187

[639 rows x 3 columns]

Version 7 of 7

Quick Version

Notebook

Input (1)

Execution Info

Log

Comments (0)

Fig 2.6: Decision Tree of Land Average Temperature (Global Input)

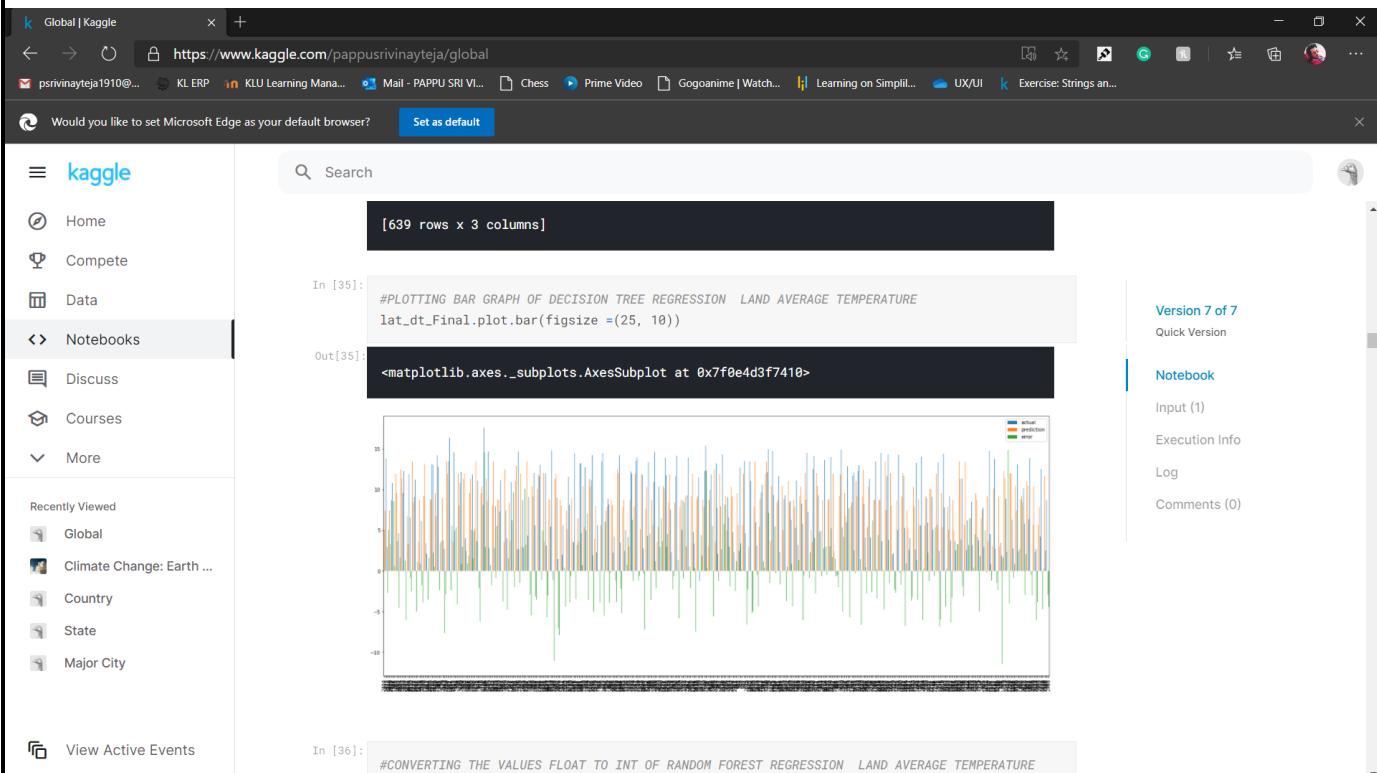


Fig 2.7: Bar Graph of Decision Tree's Land Average Temperature (Global Input)

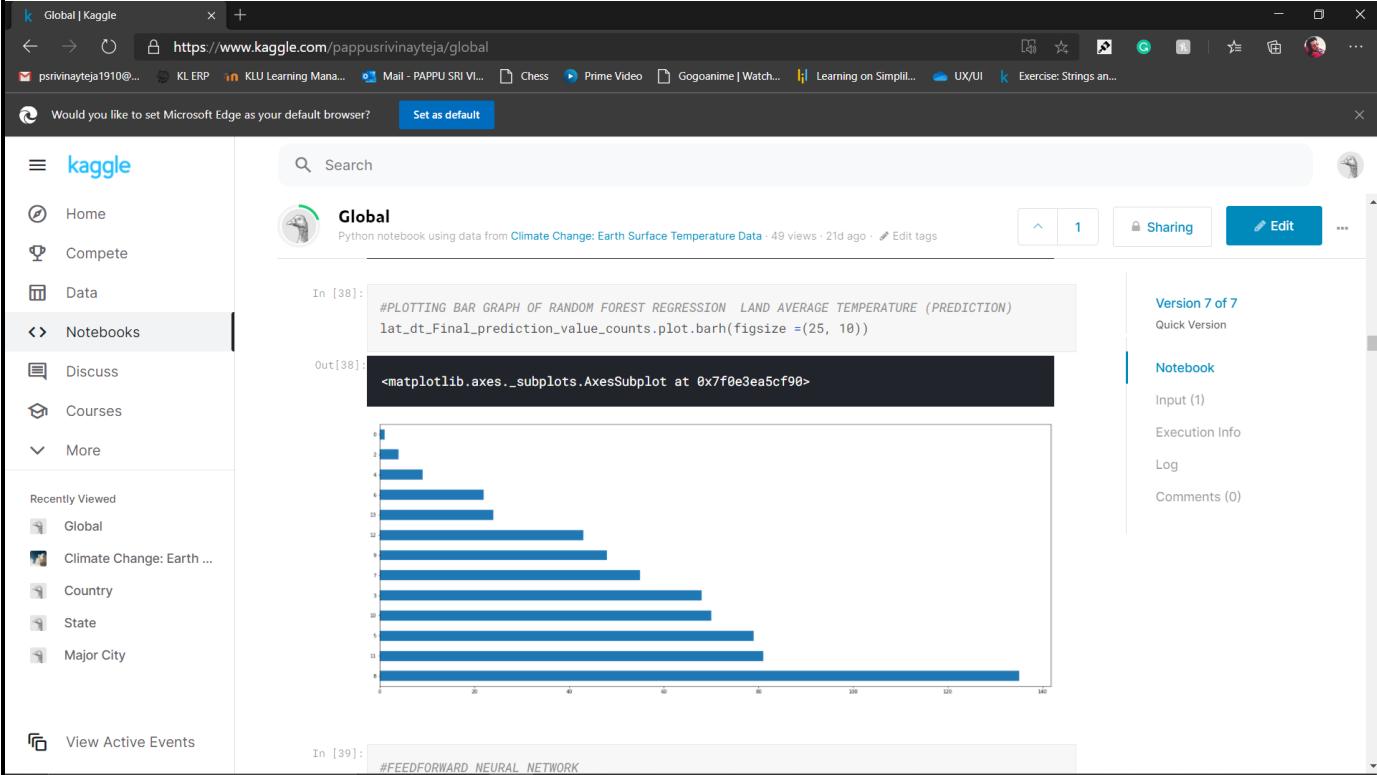


Fig 2.8: Bar Graph of highest count of Decision Tree's Land Average Temperature (Global Input)

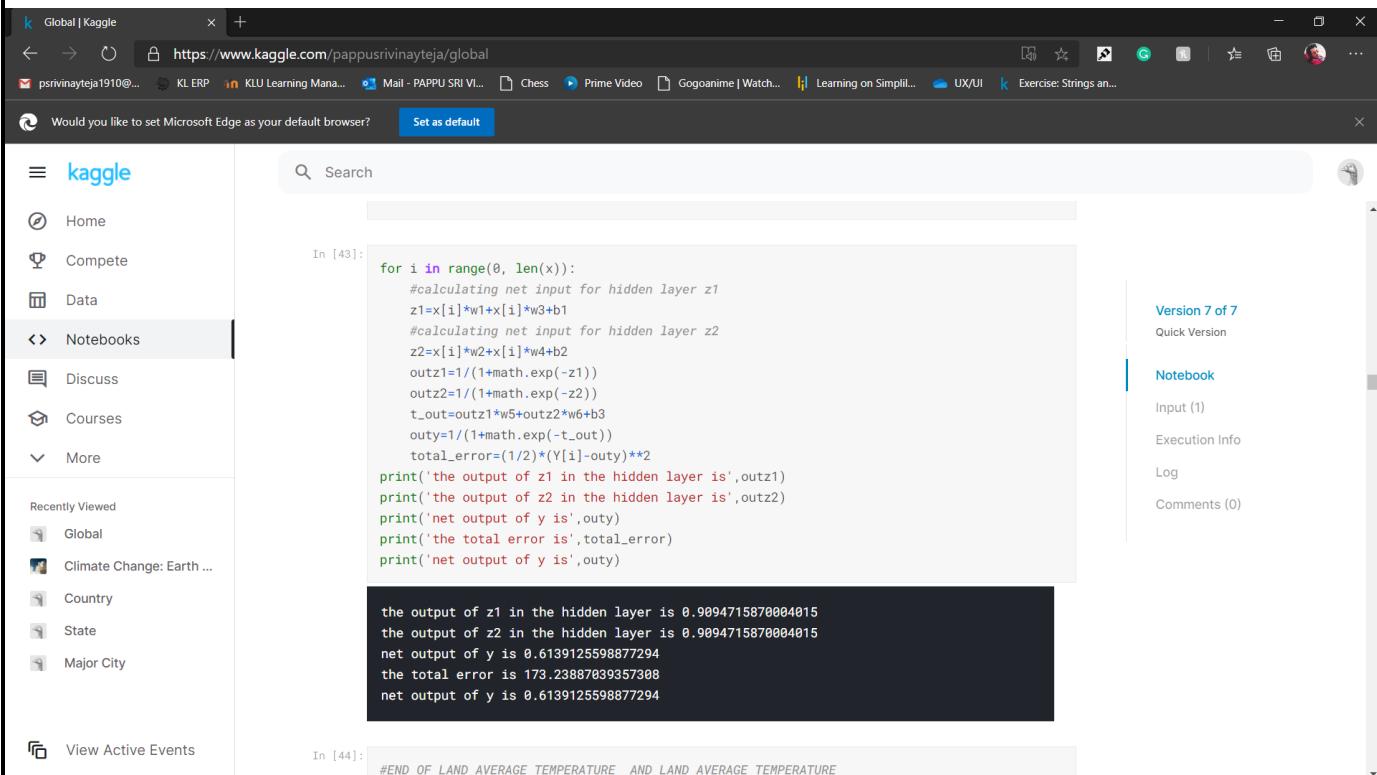


Fig 2.9: FeedForward Neural Networks of Land Average Temperature (Global Input)

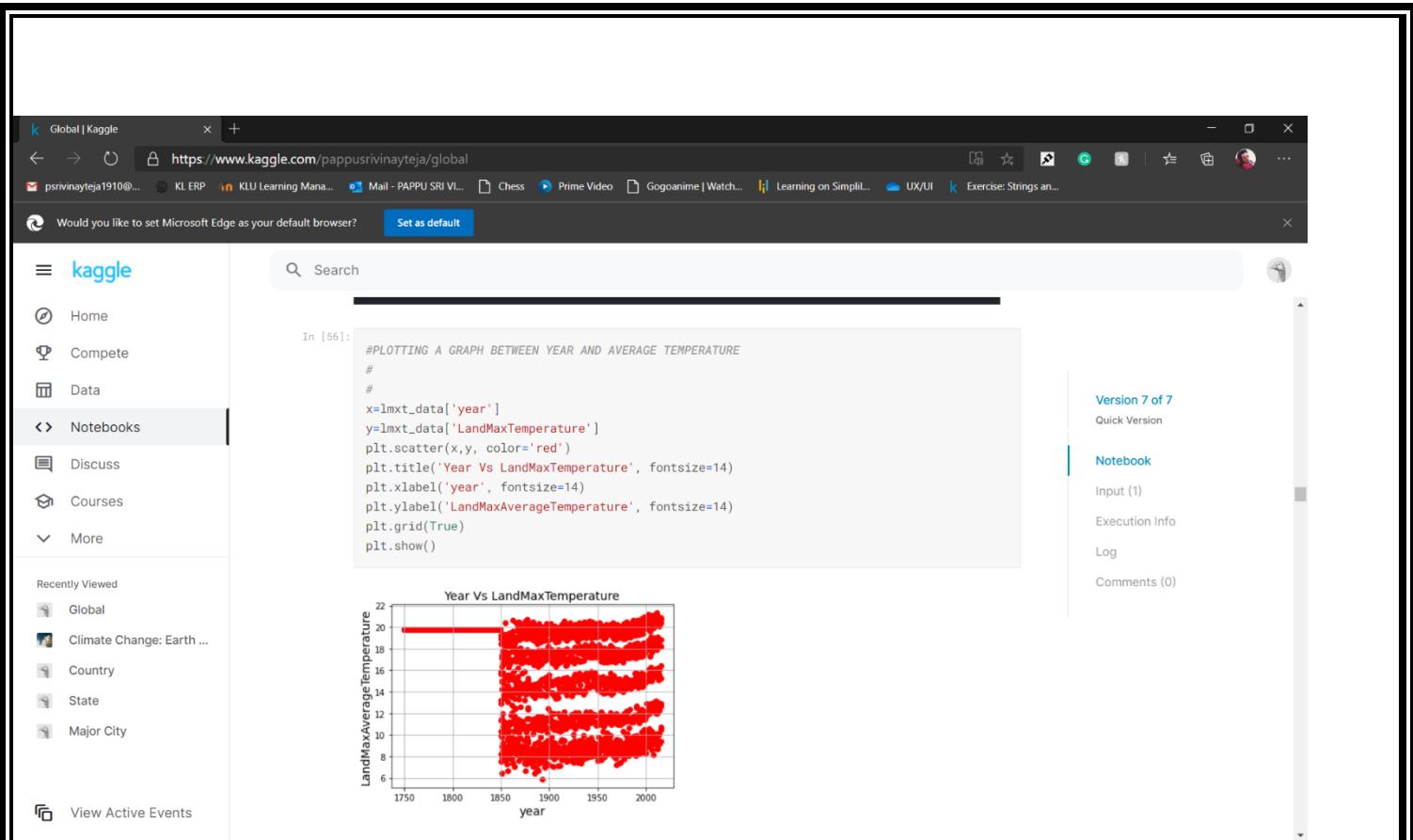


Fig 3.1: Graph of Land Max Temperature (Global Input)

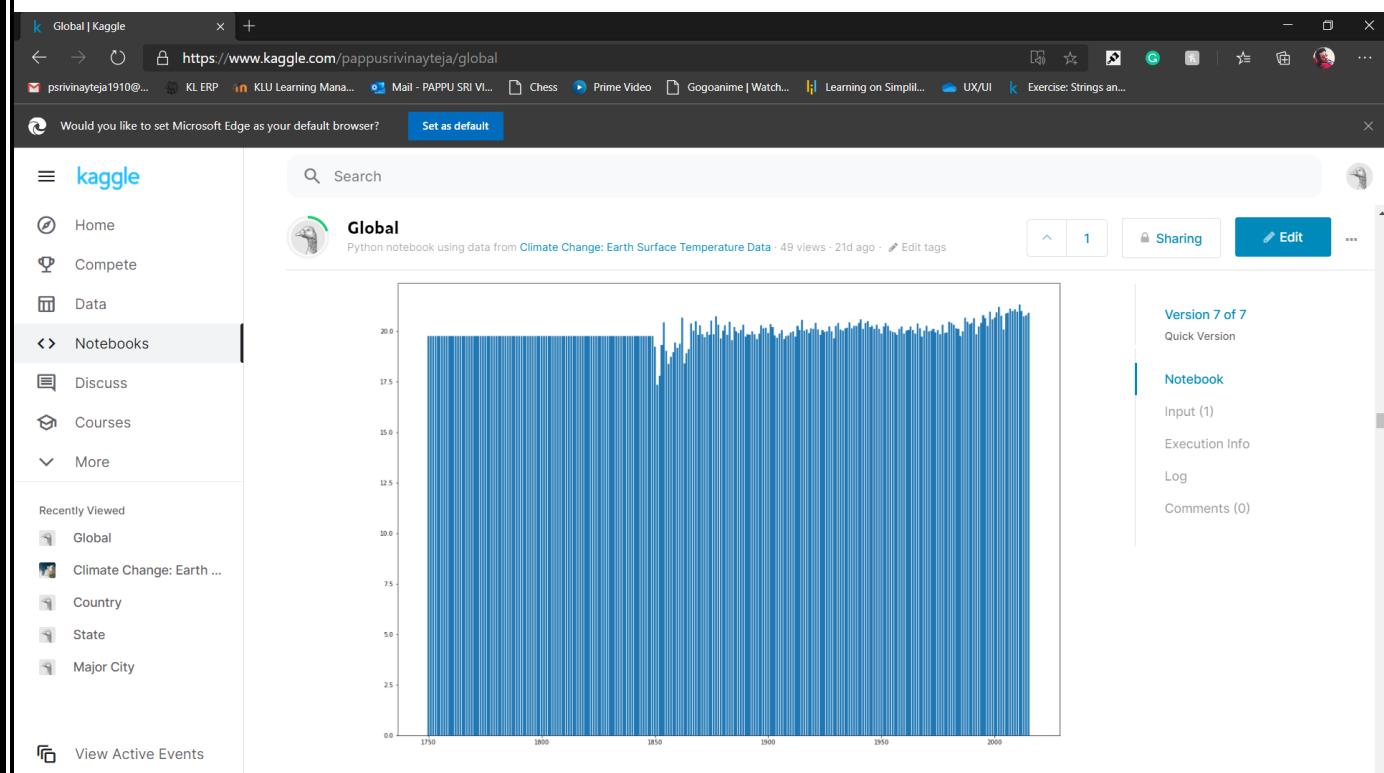


Fig 3.2: Bar Graph of Land Max Temperature (Global Input)

```

#TRAINING AND TESTING X AND y
#
#
train_X,test_X,train_y,test_y=train_test_split(X,y,test_size=0.2,random_state=4)

print('train_X.head')
print(train_X.head())

print('train_y.head')
print(train_y.head())

```

	train_X.head	train_y.head	
Unnamed: 0	year	LandMaxTemperatureUncertainty	
1804	1804	1900	0.329
3162	3162	2013	0.100
79	79	1756	0.093
1728	1728	1894	0.567
1133	1133	1844	0.093
	train_X.head	train_y.head	
	1804	17.415	
	3162	20.737	
	79	19.753	
	1728	7.410	
	1133	19.753	
	Name: LandMaxTemperature, dtype: float64	Name: LandMaxTemperature, dtype: float64	

Fig 3.3: Training and Test data of Land Max Temperature (Global Input)

```

#LINEAR REGRESSION
#
#
lr=LinearRegression()
lr.fit(train_X,train_y)

prediction = lr.predict(test_X)

lmxt_lr_Final=pd.DataFrame({'actual':test_y,'prediction' : prediction,'error':(test_y-prediction)})
print('Final linear regression predicted values')
print(lmxt_lr_Final)

```

	Final linear regression predicted values		
	actual	prediction	error
2414	11.074	14.312327	-3.238327
333	19.753	20.526852	-0.773852
1471	19.415	15.747721	3.667279
672	19.753	18.109579	1.643421
3113	20.056	13.225119	6.830881
...	...	...	...
1550	10.686	14.350326	-3.664326
2340	8.127	14.050711	-5.923711
459	19.753	19.144218	0.608782
889	19.753	17.795279	1.957721
2795	9.637	15.298922	-5.661922

Fig 3.4: Linear Regression of Land Max Temperature (Global Input)

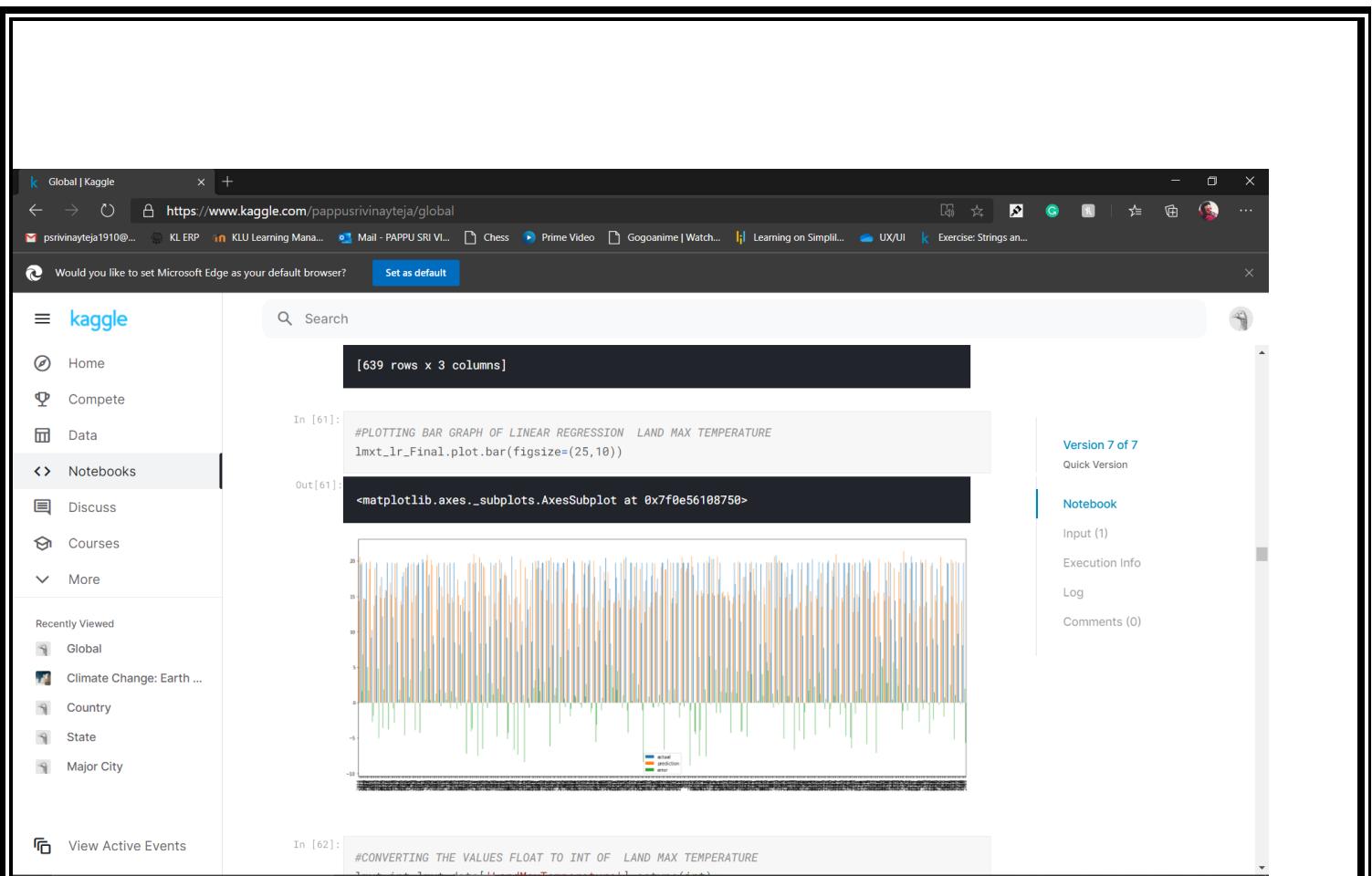


Fig 3.5: Bar Graph of Linear Regression's Land Max Temperature (Global Input)

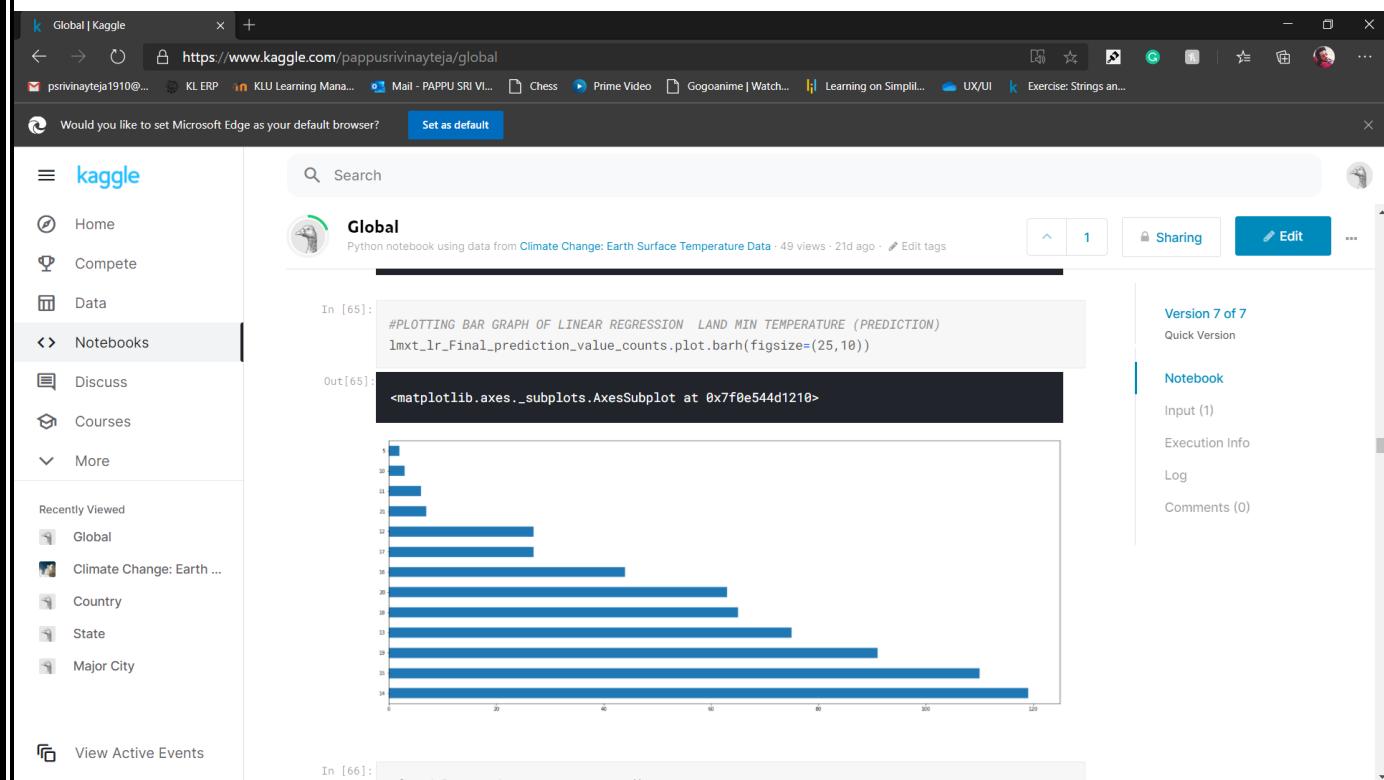


Fig 3.6: Bar Graph of highest count of Linear Regression - Land Max Temperature (Global Temperature)

The screenshot shows a Microsoft Edge browser window displaying a Kaggle notebook titled "Global". The notebook interface includes a sidebar with navigation links like Home, Compete, Data, Notebooks, Discuss, Courses, and More. The main area contains a search bar and a code editor with the following content:

```

rf_model = RandomForestRegressor()

# fit your model
rf_model.fit(train_X, train_y)

# Calculate the mean absolute error of your Random Forest model on the validation data
rf_prediction = rf_model.predict(test_X)

lmxt_rf_Final=pd.DataFrame({'actual':test_y,'prediction' : rf_prediction,'error':(test_y-rf_prediction)})
print(lmxt_rf_Final)

```

Below the code, the notebook displays a table of data:

	actual	prediction	error
2414	11.074	13.75541	-2.681410e+00
333	19.753	19.75300	-1.172396e-13
1471	19.415	18.88228	5.327280e-01
672	19.753	19.75300	-1.172396e-13
3113	20.056	17.73587	2.320130e+00
...	...	...	...
1550	10.686	11.12150	-4.355000e-01
2340	8.127	10.99518	-2.868180e+00
459	19.753	19.75300	-1.172396e-13
889	19.753	19.75300	-1.172396e-13
2795	9.637	10.60622	-9.692200e-01

On the right side of the notebook, there are sections for "Version 7 of 7", "Notebook", and "Input (1)".

Fig 3.7: Random Forest Regression of Land Max Temperature (Global Input)

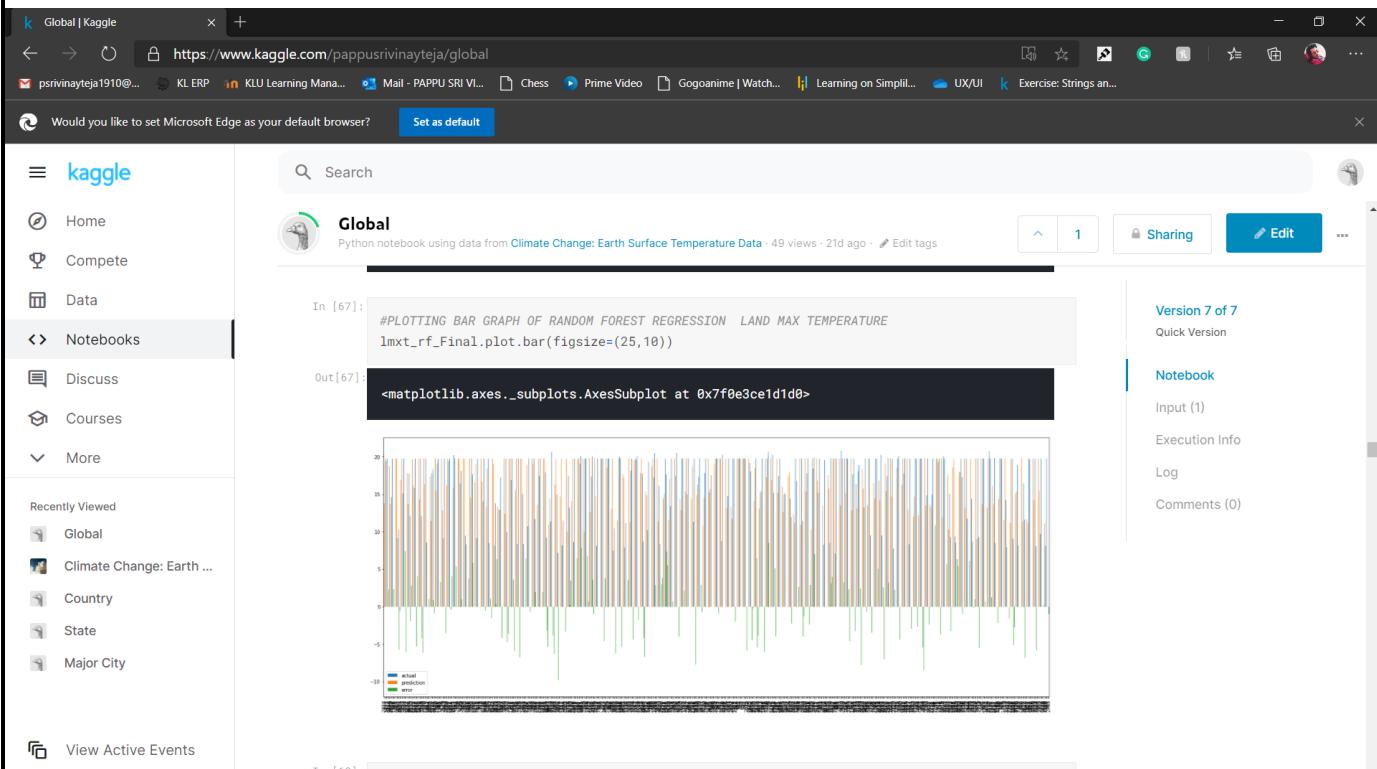


Fig 3.8: Bar Graph of Random Forest's Land Max Temperature (Global Input)

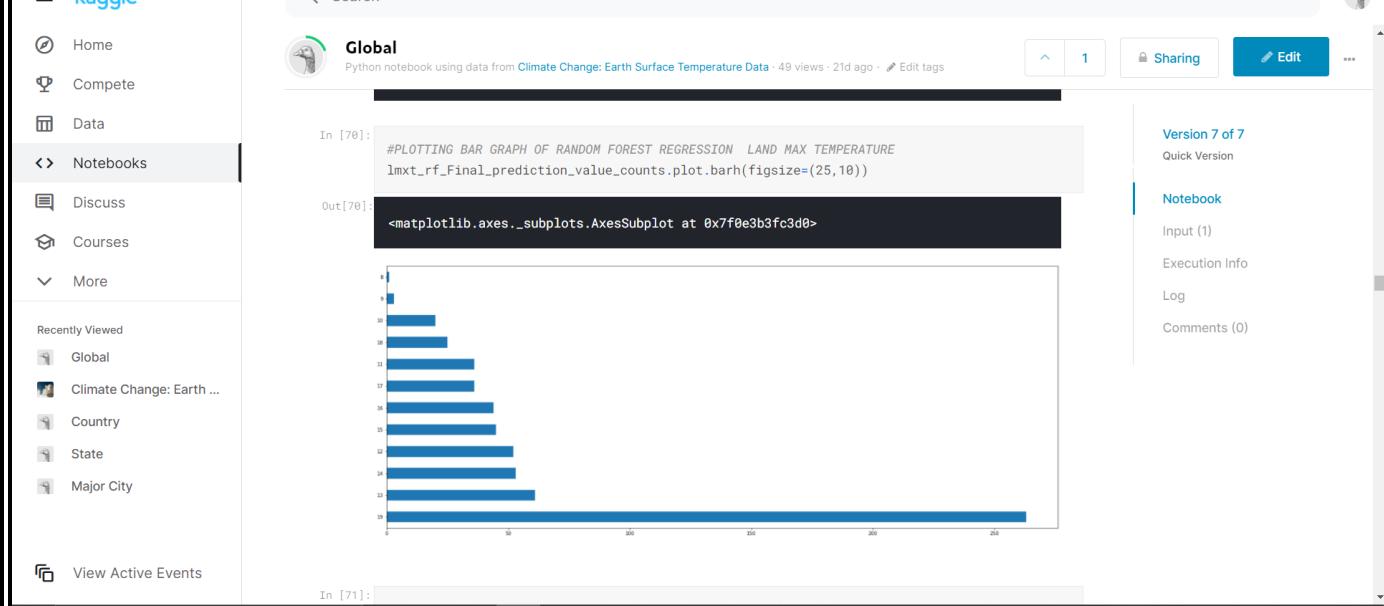


Fig 3.9: Bar Graph of Random Forest’s Land Max Temperature (Global Input)

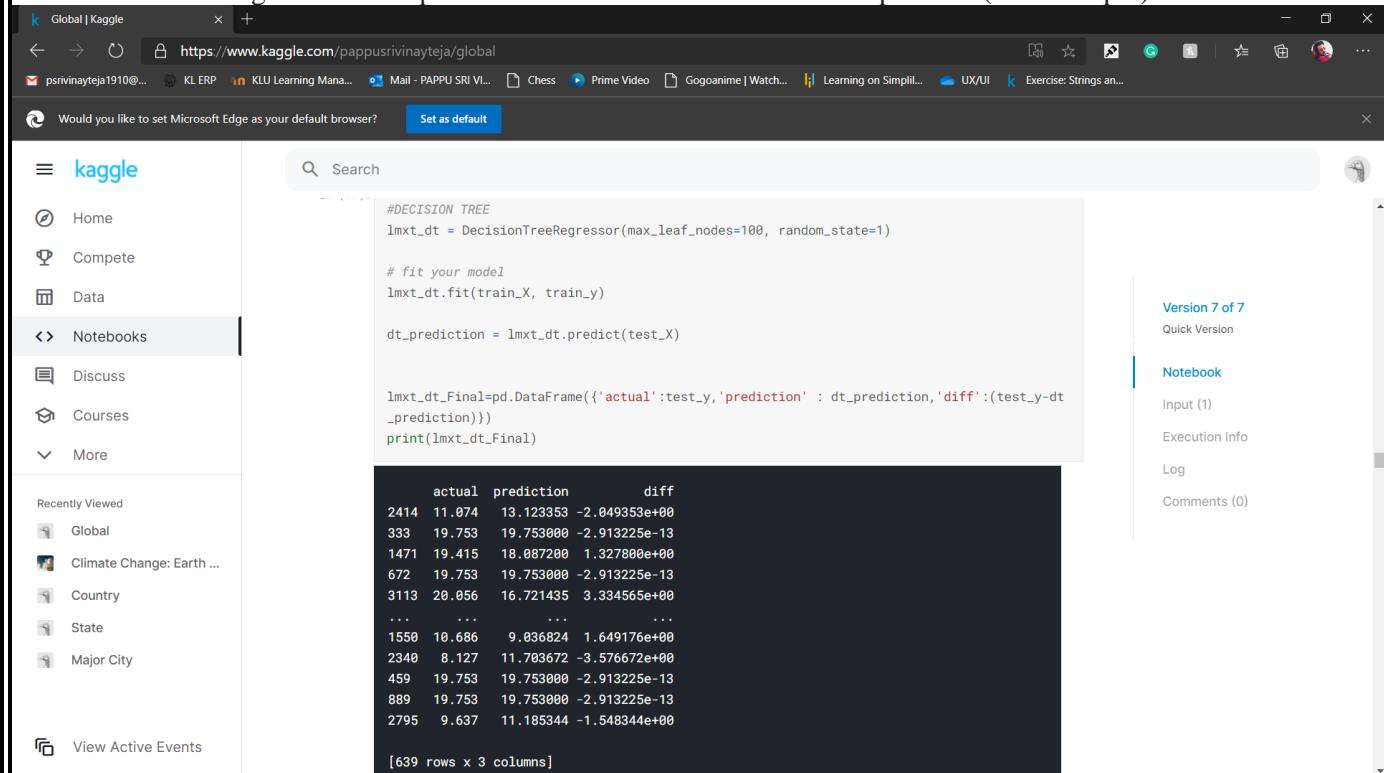


Fig 3.10: Decision Tree of Land Max Temperature (Global Input)

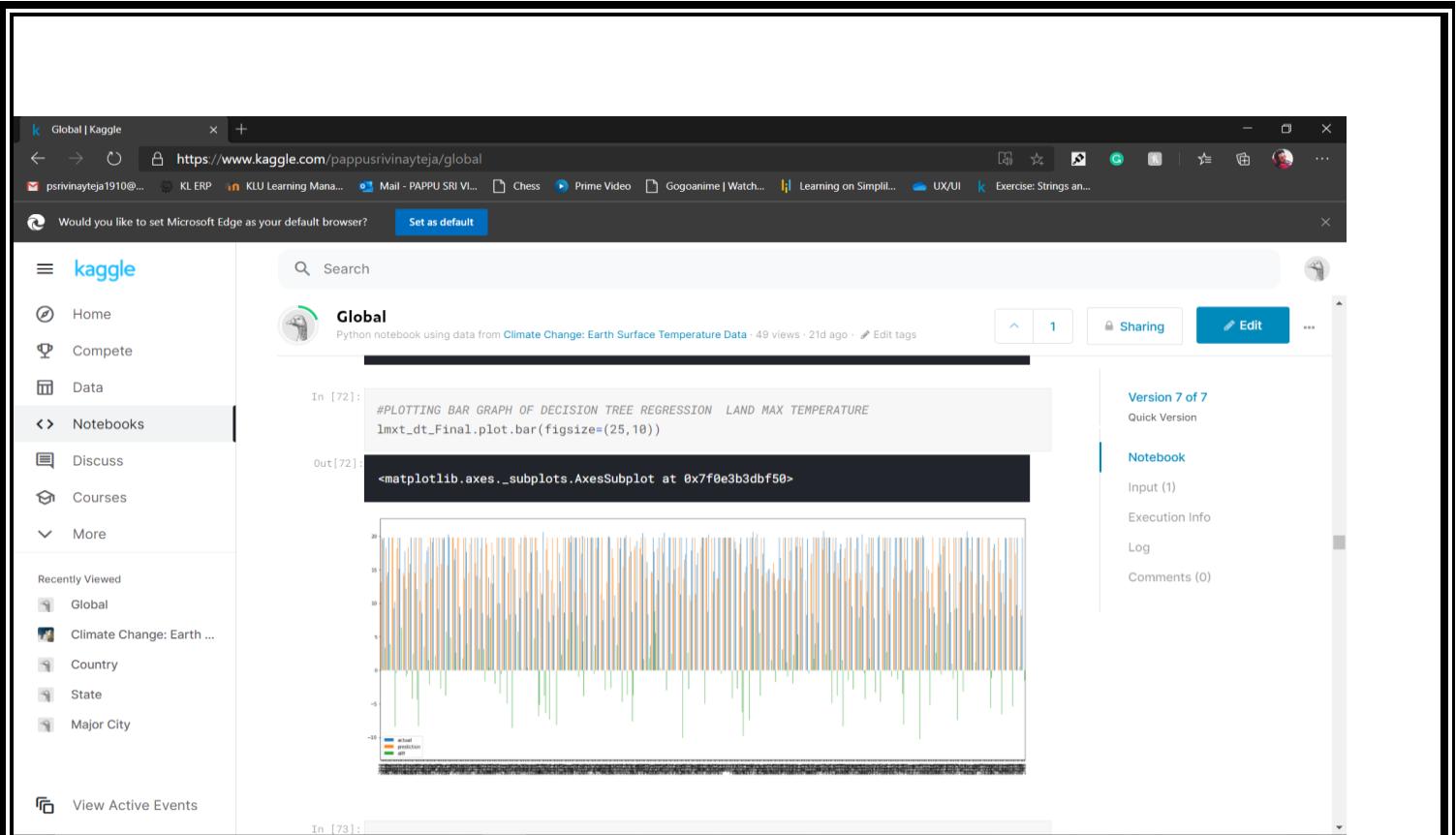


Fig 3.11: Bar Graph of Decision Tree's Land Max Temperature (Global Input)

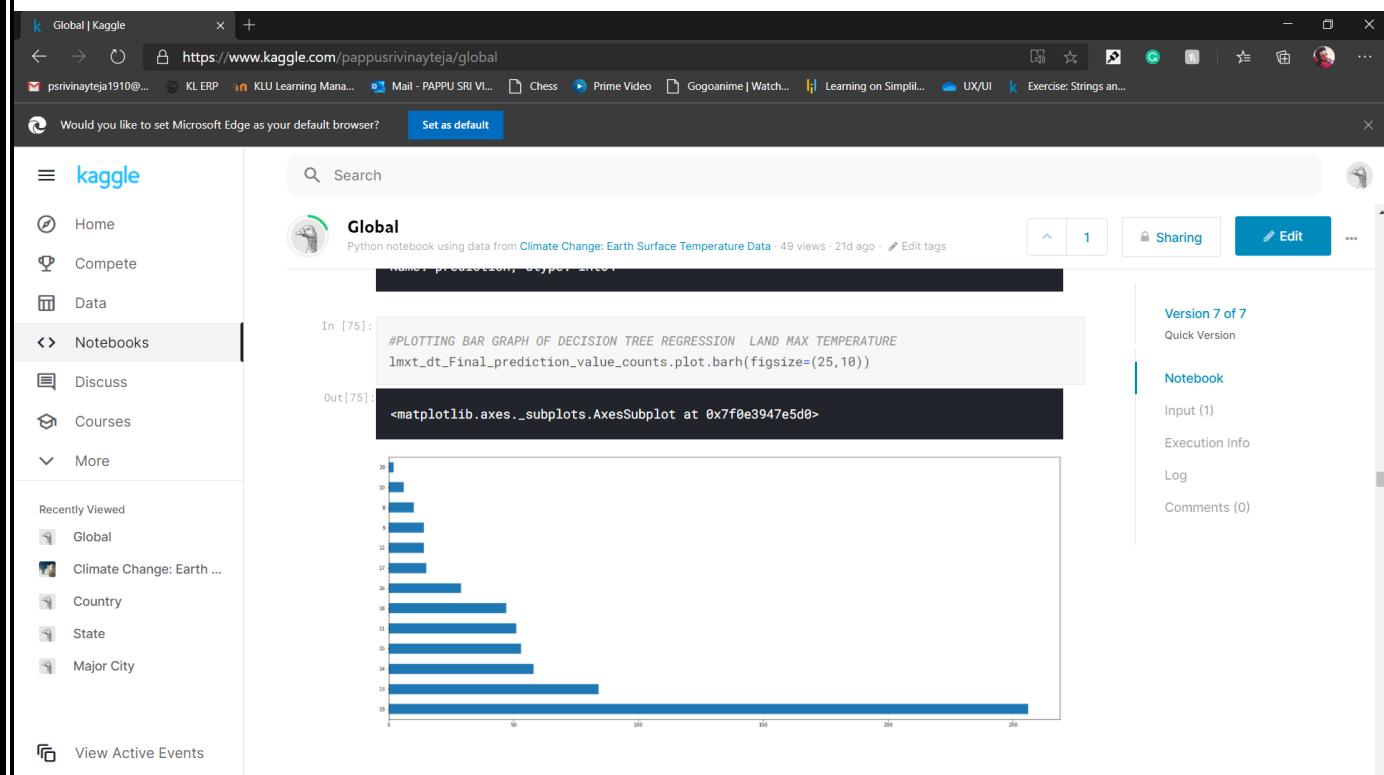


Fig 3.12: Bar Graph of highest count of Decision Tree's Land Max Temperature (Global Input)

k Global | Kaggle + https://www.kaggle.com/pappusirivinayteja/global

Would you like to set Microsoft Edge as your default browser? Set as default

**kaggle**

- Home
- Compete
- Data
- Notebooks
- Discuss
- Courses
- More

Recently Viewed

- Global
- Climate Change: Earth ...
- Country
- State
- Major City

View Active Events

Search

```
In [80]: for i in range(0, len(x)):
    #calculating net input for hidden layer z1
    z1=x[i]*W1*x[i]*w3+b1
    #calculating net input for hidden layer z2
    z2=x[i]*W2*x[i]*w4+b2
    outz1=1/(1+math.exp(-z1))
    outz2=1/(1+math.exp(-z2))
    t_out=outz1*w5+outz2*w6+b3
    outy=1/(1+math.exp(-t_out))
    total_error=(1/2)*(y[i]-outy)**2
print('the output of z1 in the hidden layer is',outz1)
print('the output of z2 in the hidden layer is',outz2)
print('net output of y is',outy)
print('the total error is',total_error)
print('net output of y is',outy)

the output of z1 in the hidden layer is 0.9877511652624036
the output of z2 in the hidden layer is 0.9877511652624036
net output of y is 0.6213072336019416
the total error is 118.25209560163199
net output of y is 0.6213072336019416
```

In [81]: #END OF LAND MAX TEMPERATURE AND LAND MAX TEMPERATURE UNCERTAINTY

Fig 3.13: Feed Forward Neural Networks of Land Average Temperature (Global Input)

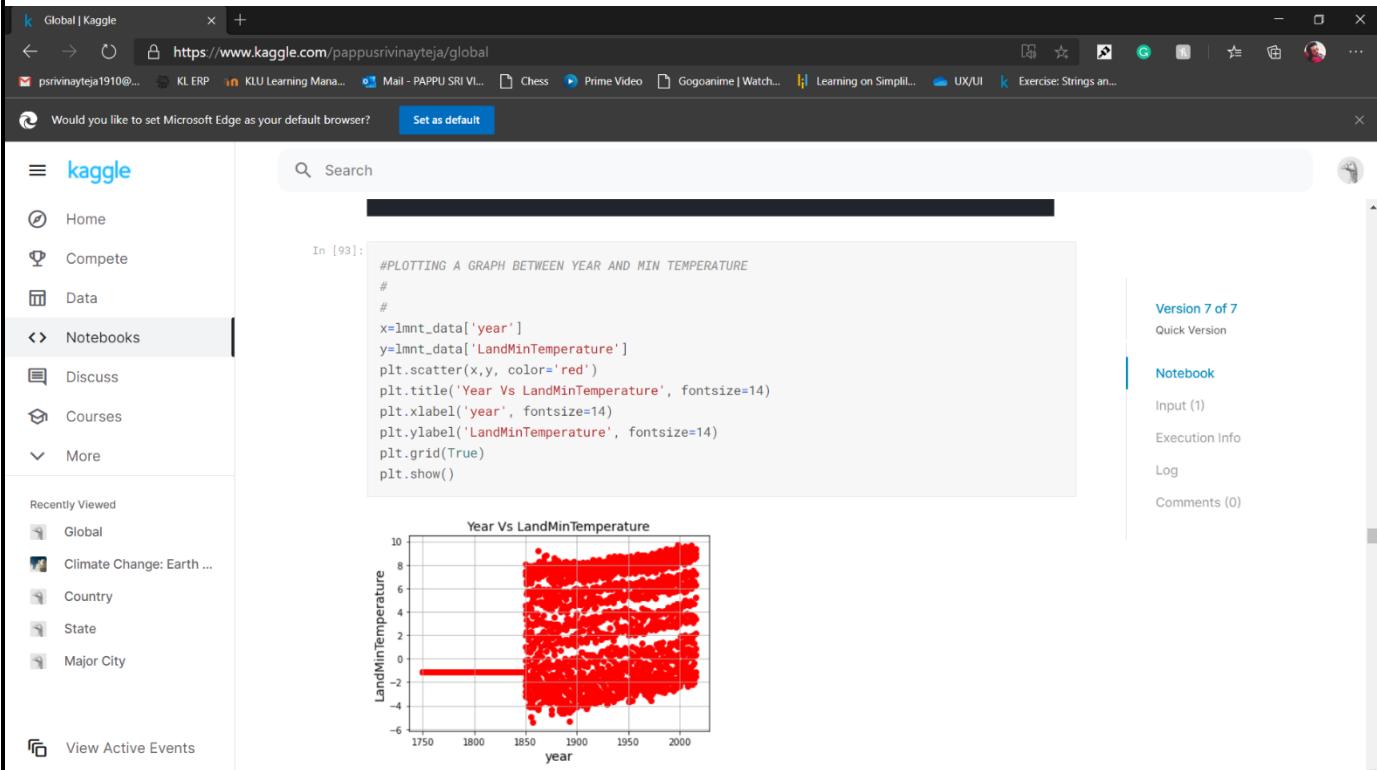


Fig 4.1: Graph of Land Min Temperature (Global Input)

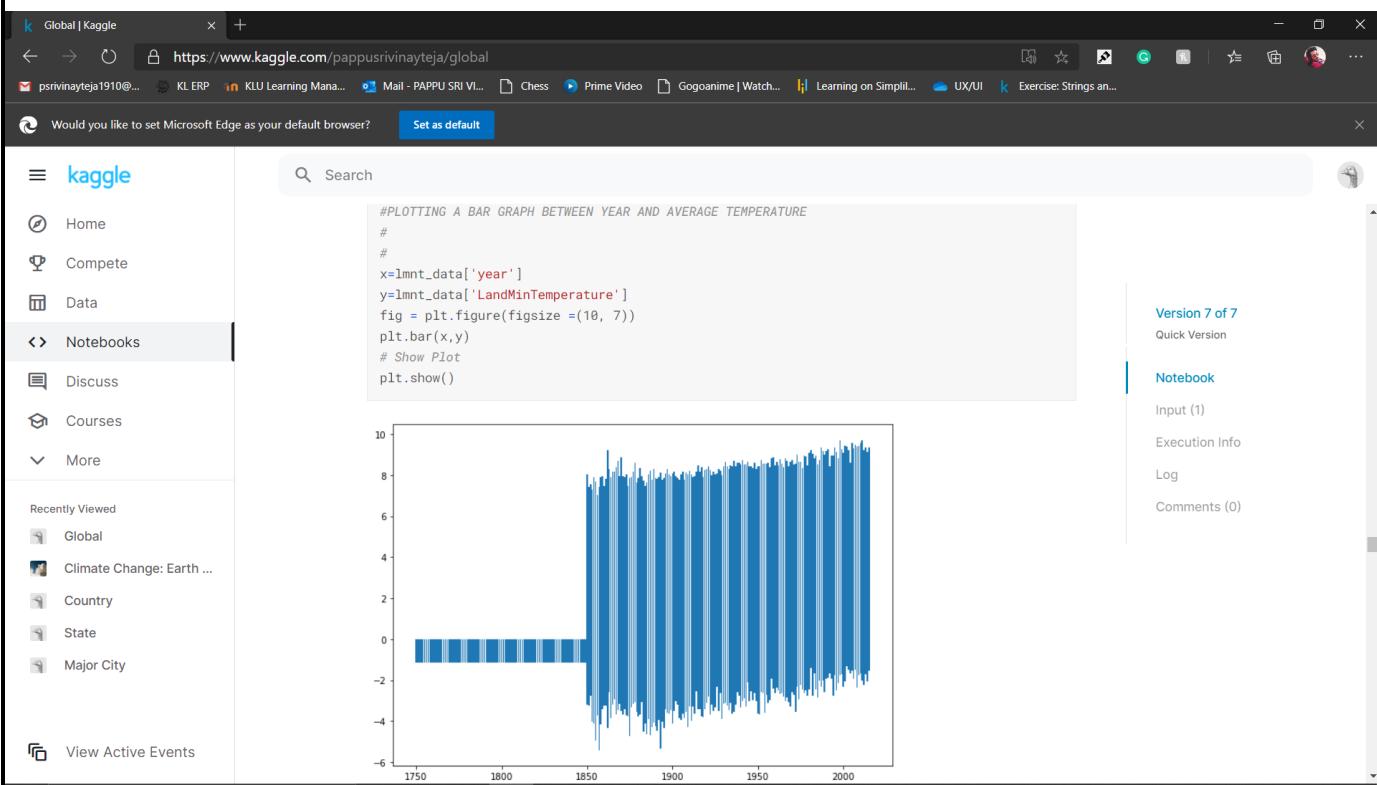


Fig 4.2: Bar Graph of Land Min Temperature (Global Input)

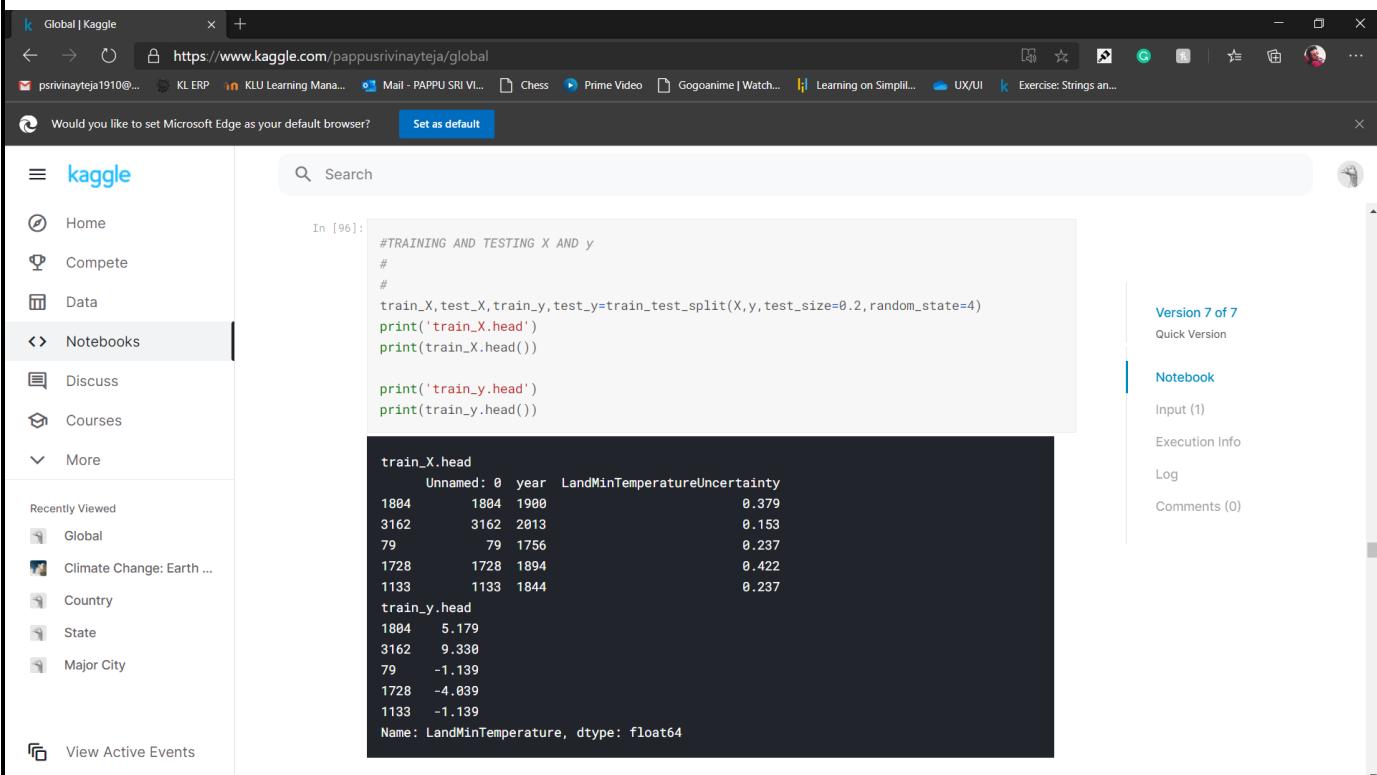


Fig 4.3: Training and Test data of Land Min Temperature (Global Input)

The screenshot shows a Microsoft Edge browser window displaying a Kaggle notebook titled "Global". The notebook contains Python code for linear regression:

```

#LINEAR REGRESSION
#
#
lr=LinearRegression()
lr.fit(train_X,train_y)

prediction = lr.predict(test_X)

lmnt_lr_Final=pd.DataFrame({'actual':test_y,'prediction' : prediction,'diff':(test_y-prediction)})
print('Final Linear regression predicted values')
print(lmnt_lr_Final)

```

The output cell displays the "Final Linear regression predicted values" as a DataFrame:

	actual	prediction	diff
2414	-1.087	2.852734	-3.139734
333	-1.139	-0.520409	-0.618591
1471	7.451	1.720924	5.738076
672	-1.139	-1.762716	0.623716
3113	8.300	4.037083	4.262917
...	...	...	...
1558	-1.389	0.662216	-2.051216
2340	-2.987	1.511147	-4.418147
459	-1.139	-1.545128	0.406128
889	-1.139	-1.124226	-0.014774
2795	-0.705	4.655554	-5.360554

Fig 4.4: Linear Regression of Land Min Temperature (Global Input)

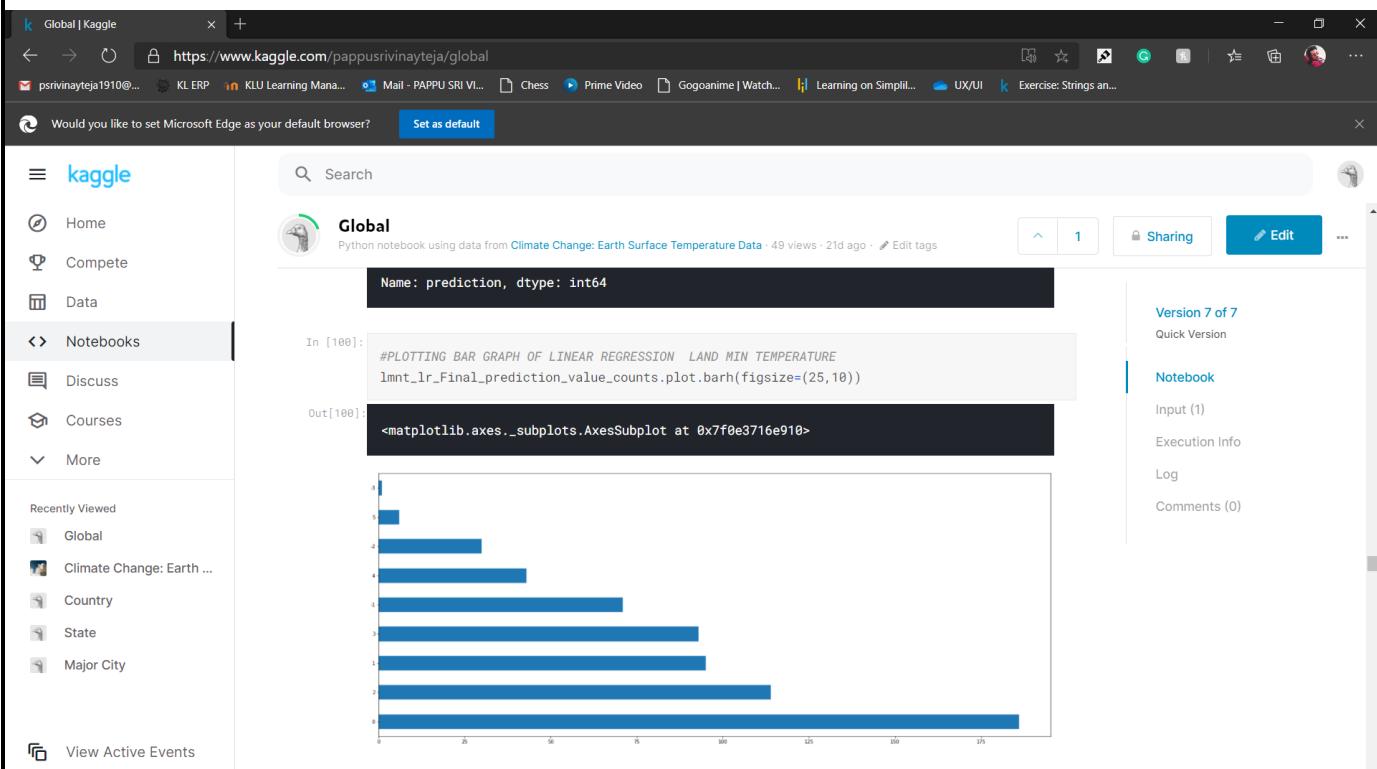


Fig 4.5: Bar Graph of Linear Regression's Land Min Temperature (Global Input)

Global | Kaggle

Would you like to set Microsoft Edge as your default browser? Set as default

**kaggle**

Search

Home Compete Data Notebooks (selected) Discuss Courses More

Recently Viewed Global Climate Change: Earth ... Country State Major City View Active Events

```

rf_model = RandomForestRegressor()

# fit your model
rf_model.fit(train_X, train_y)

# Calculate the mean absolute error of your Random Forest model on the validation data
rf_prediction = rf_model.predict(test_X)

lmnt_rf_Final=pd.DataFrame({'actual':test_y,'prediction' : rf_prediction,'diff':(test_y-rf_prediction)})
print(lmnt_rf_Final)

```

	actual	prediction	diff
2414	-1.087	-0.97555	-1.114500e-01
333	-1.139	-1.13900	6.439294e-15
1471	7.451	1.98257	5.468430e+00
672	-1.139	-1.13900	6.661338e-15
3113	8.300	4.64847	3.651530e+00
...	...	...	...
1550	-1.389	2.63703	-4.026030e+00
2340	-2.907	3.12761	-6.034610e+00
459	-1.139	-1.13900	6.661338e-15
889	-1.139	-1.13900	6.661338e-15
2795	-0.705	-0.02043	-6.845700e-15

Version 7 of 7 Quick Version

Notebook Input (1) Execution Info Log Comments (0)

Fig 4.6: Random Forest Regression of Land Min Temperature (Global Input)

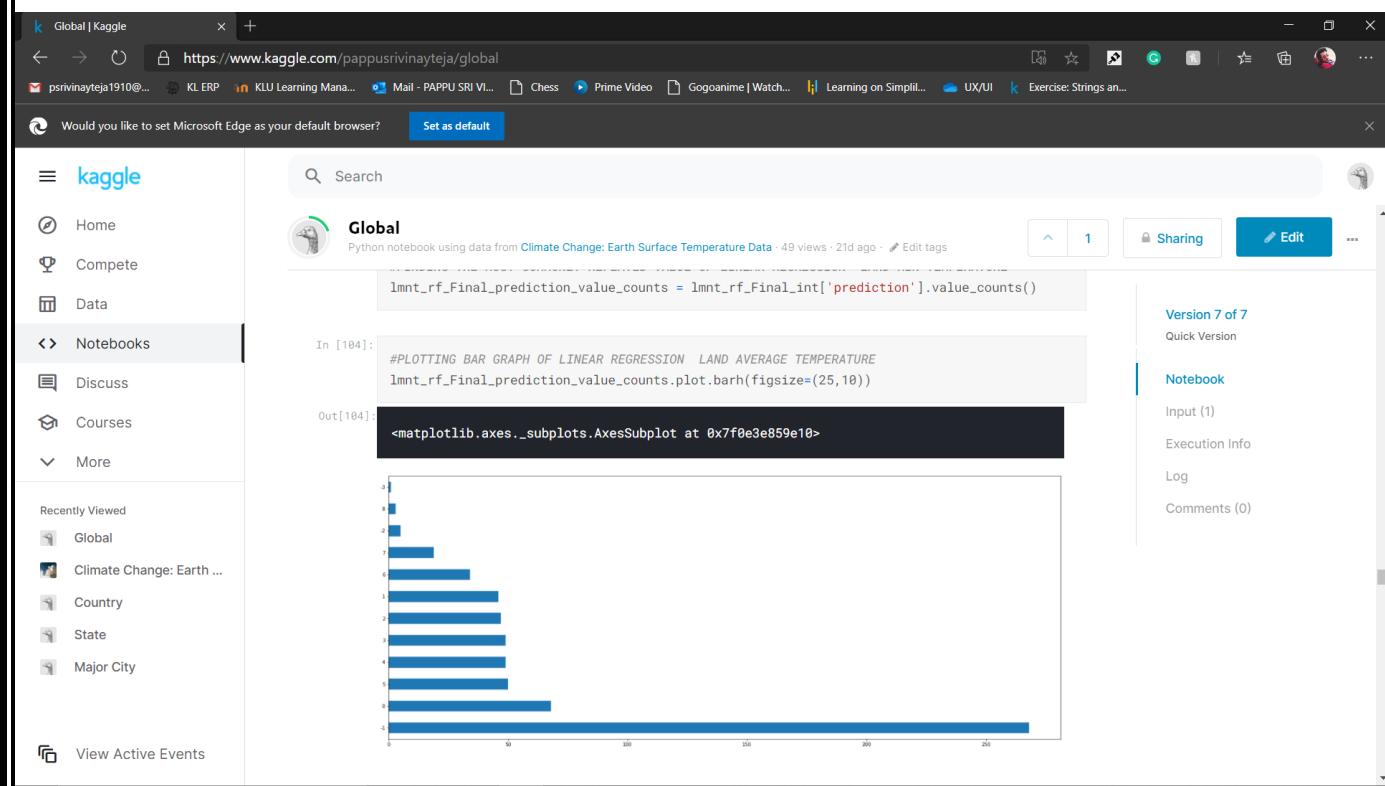


Fig 4.8: Bar Graph of highest count of Random Forest - Land Min Temperature (Global Temperature)

```
#DECISION TREE
dt = DecisionTreeRegressor(max_leaf_nodes=100, random_state=1)

# fit your model
dt.fit(train_X, train_y)

lmnt_dt_prediction = dt.predict(test_X)

lmnt_dt_Final=pd.DataFrame({'actual':test_y,'prediction' : lmnt_dt_prediction,'diff':(test_y-lmnt_dt_prediction)})
print(lmnt_dt_Final)

      actual   prediction      diff
2414 -1.087    -1.087000  0.000000
333  -1.139    -1.143199  0.004199
1471  7.451     -1.496526  8.947526
672  -1.139    -1.143199  0.004199
3113  8.300     4.608111  3.691889
...       ...
1550 -1.389    1.548280 -2.937280
2340 -2.907    1.352120 -4.259120
459  -1.139    -1.143199  0.004199
889  -1.139    -1.143199  0.004199
2795 -0.785     0.341717 -1.1046717

[639 rows x 3 columns]
```

Fig 4.9: Decision Tree of Land Average Temperature (Global Input)

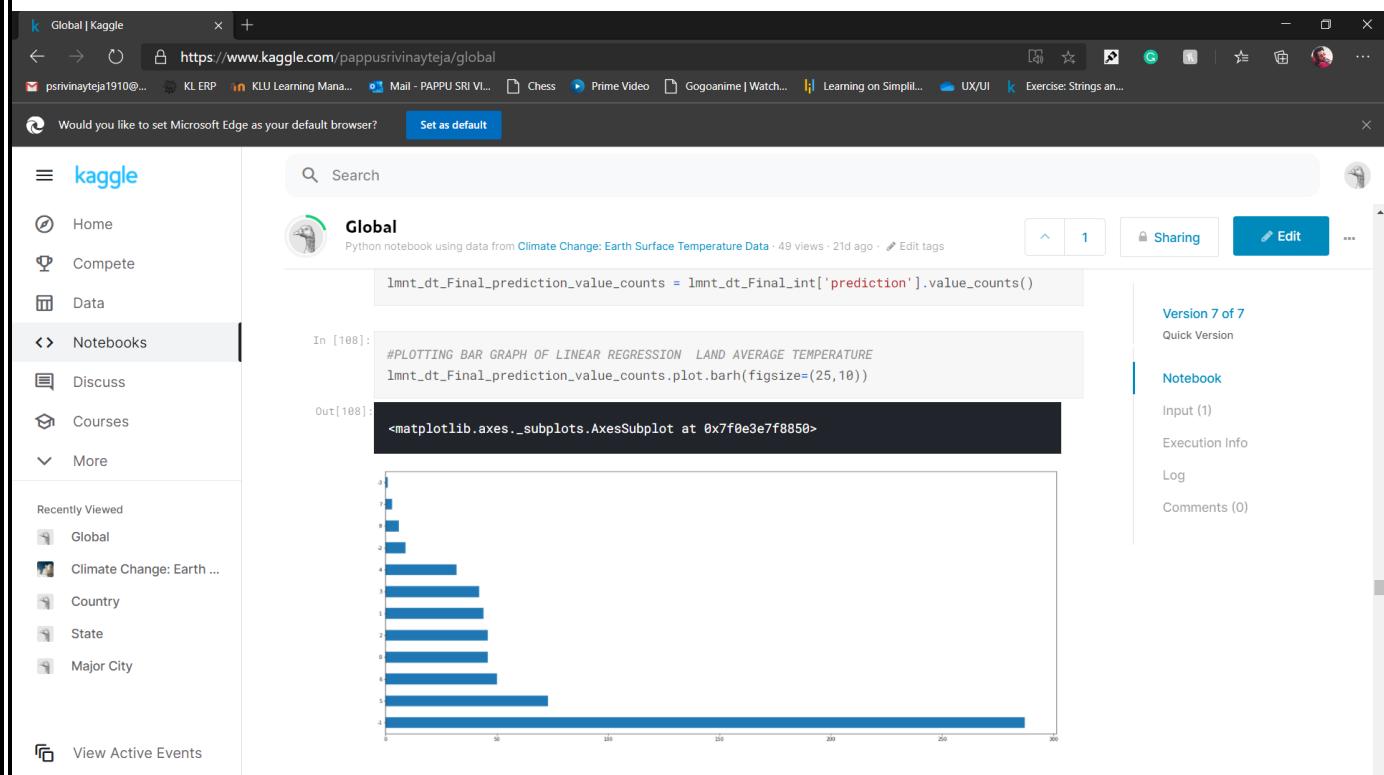


Fig 4.10: Bar Graph of highest count of Decision Tree's Land Min Temperature (Global Input)

```

for i in range(0, len(x)):
    #calculating net input for hidden layer z1
    z1=x[i]*w1+x[i]*w3+b1
    #calculating net input for hidden layer z2
    z2=x[i]*w2+x[i]*w4+b2
    outz1=1/(1+math.exp(-z1))
    outz2=1/(1+math.exp(-z2))
    t_out=outz1*w5+outz2*w6+b3
    outy=1/(1+math.exp(-t_out))
    total_error=(1/2)*(y[i]-outy)**2
print('the output of z1 in the hidden layer is',outz1)
print('the output of z2 in the hidden layer is',outz2)
print('net output of y is',outy)
print('the total error is',total_error)
print('net output of y is',outy)

the output of z1 in the hidden layer is 0.5534944760118916
the output of z2 in the hidden layer is 0.5534944760118916
net output of y is 0.579664866559352
the total error is 1124.3440920839125
net output of y is 0.579664866559352

```

Fig 4.11: Feed Forward Neural Networks of Land Min Temperature (Global Input)

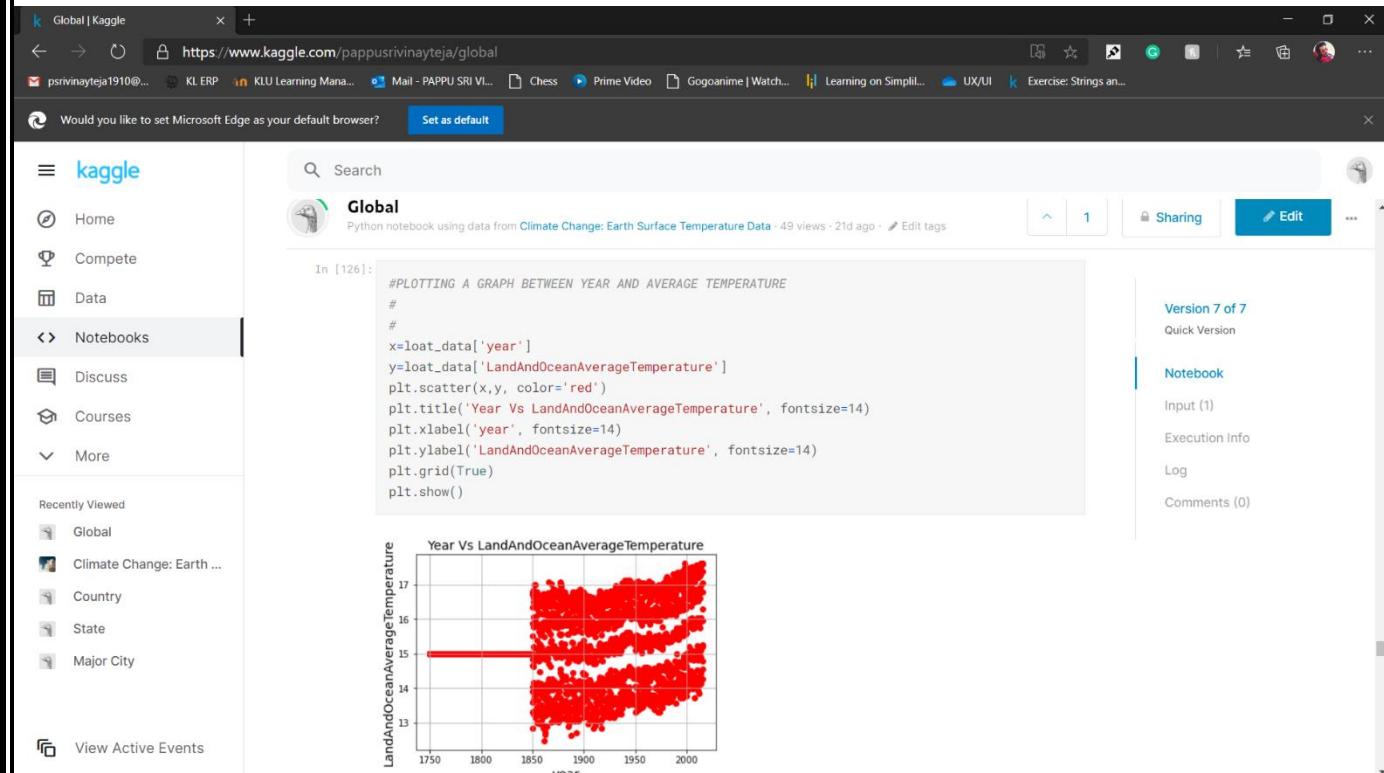


Fig 5.1: Graph of Land And Ocean Temperature (Global Input)

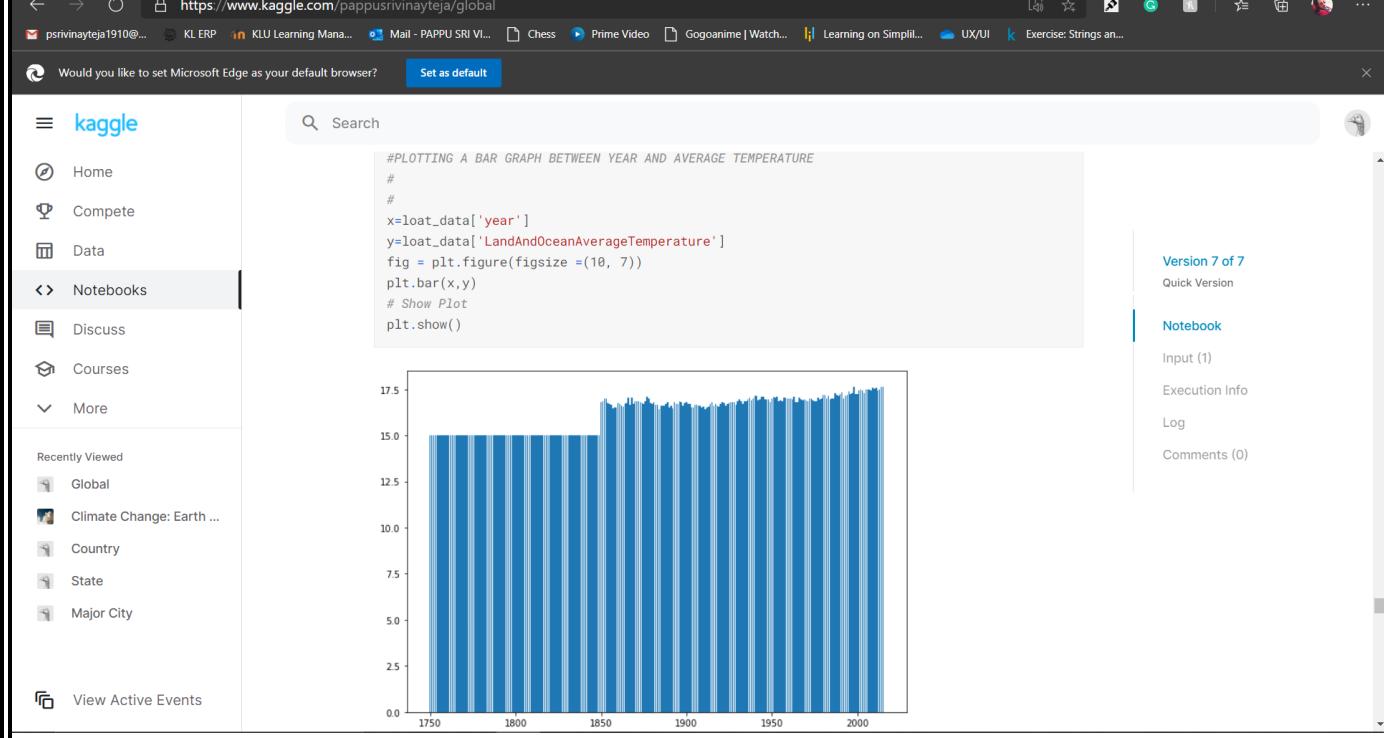


Fig 5.2: Bar Graph of Land And Ocean Temperature (Global Input)

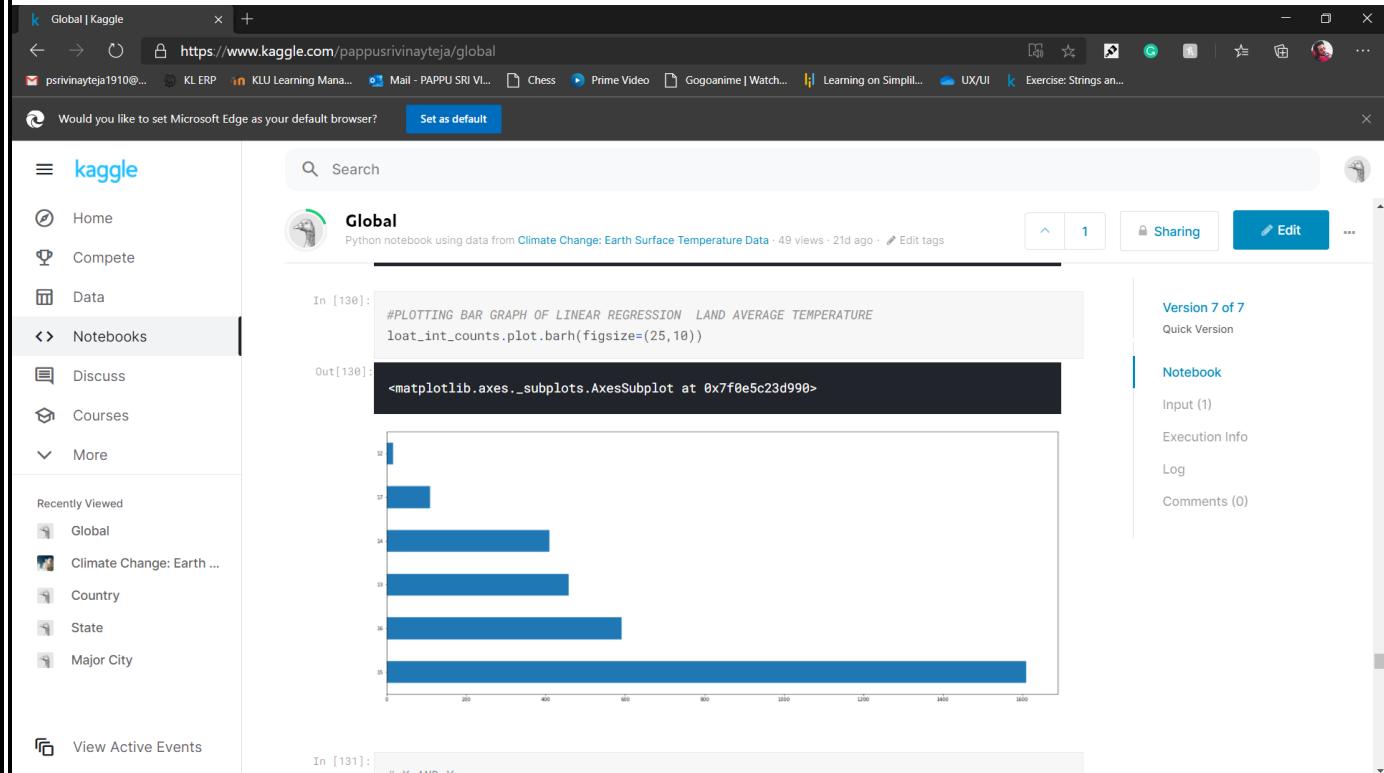


Fig 5.3: Bar Graph of Land Max Temperature (Global Input)

The screenshot shows a Microsoft Edge browser window displaying a Kaggle notebook titled "Global". The notebook URL is <https://www.kaggle.com/pappusrinivayteja/global>. The left sidebar shows navigation links like Home, Compete, Data, Notebooks, Discuss, Courses, and More. The "Notebooks" link is currently selected. The main area displays Python code in a code editor. The code includes importing pandas, splitting the data into train\_X, test\_X, train\_y, and test\_y, and then printing the head of train\_X and train\_y. The output shows the first few rows of the data frames.

```

#TRAINING AND TESTING X AND y
train_X,test_X,train_y,test_y=train_test_split(X,y,test_size=0.2,random_state=4)
print('train_X.head')
print(train_X.head())
print('train_y.head')
print(train_y.head())

train_X.head
      Unnamed: 0  year  LandAndOceanAverageTemperatureUncertainty
1884       1884  1900                  0.159
3162       3162  2013                  0.068
79          79   1756                  0.061
1728       1728  1894                  0.159
1133       1133  1844                  0.061

train_y.head
      1884    15.955
      3162    17.503
      79     15.005
     1728    13.039
     1133    15.005
Name: LandAndOceanAverageTemperature, dtype: float64

```

Fig 5.4: Training and Test data of Land and Ocean Temperature (Global Input)

The screenshot shows a Microsoft Edge browser window displaying a Kaggle notebook titled "#LINEAR REGRESSION". The notebook URL is <https://www.kaggle.com/pappusrinivayteja/global>. The left sidebar shows navigation links like Home, Compete, Data, Notebooks, Discuss, Courses, and More. The "Notebooks" link is currently selected. The main area displays Python code in a code editor. The code imports LinearRegression from sklearn, fits it to the training data, makes predictions on the test data, and prints the final linear regression predicted values. The output shows a DataFrame with columns: actual, prediction, and error.

```

#LINEAR REGRESSION

lr=LinearRegression()
lr.fit(train_X,train_y)

loat_lr_prediction = lr.predict(test_X)

loat_lr_Final=pd.DataFrame({'actual':test_y,'prediction' : loat_lr_prediction,'error':(test_y-loat_lr_prediction)})
print('Final Linear regression predicted values')
print(loat_lr_Final)

Final Linear regression predicted values
      actual  prediction    error
2414  14.278  15.153252 -0.875252
333   15.005  15.117516 -0.112516
1471  16.822  14.980467  1.841533
672   15.005  14.778470  0.226530
3113  17.268  15.495762  1.764238
...
      ...
      ...
1550  14.340  14.883744 -0.463744
2340  13.586  14.989541 -1.403541
459   15.005  14.871667  0.133333
889   15.005  14.866613  0.138387
2795  14.068  15.701600 -1.633600
[639 rows x 3 columns]

```

Fig 5.5: Linear Regression of Land and Ocean Temperature (Global Input)

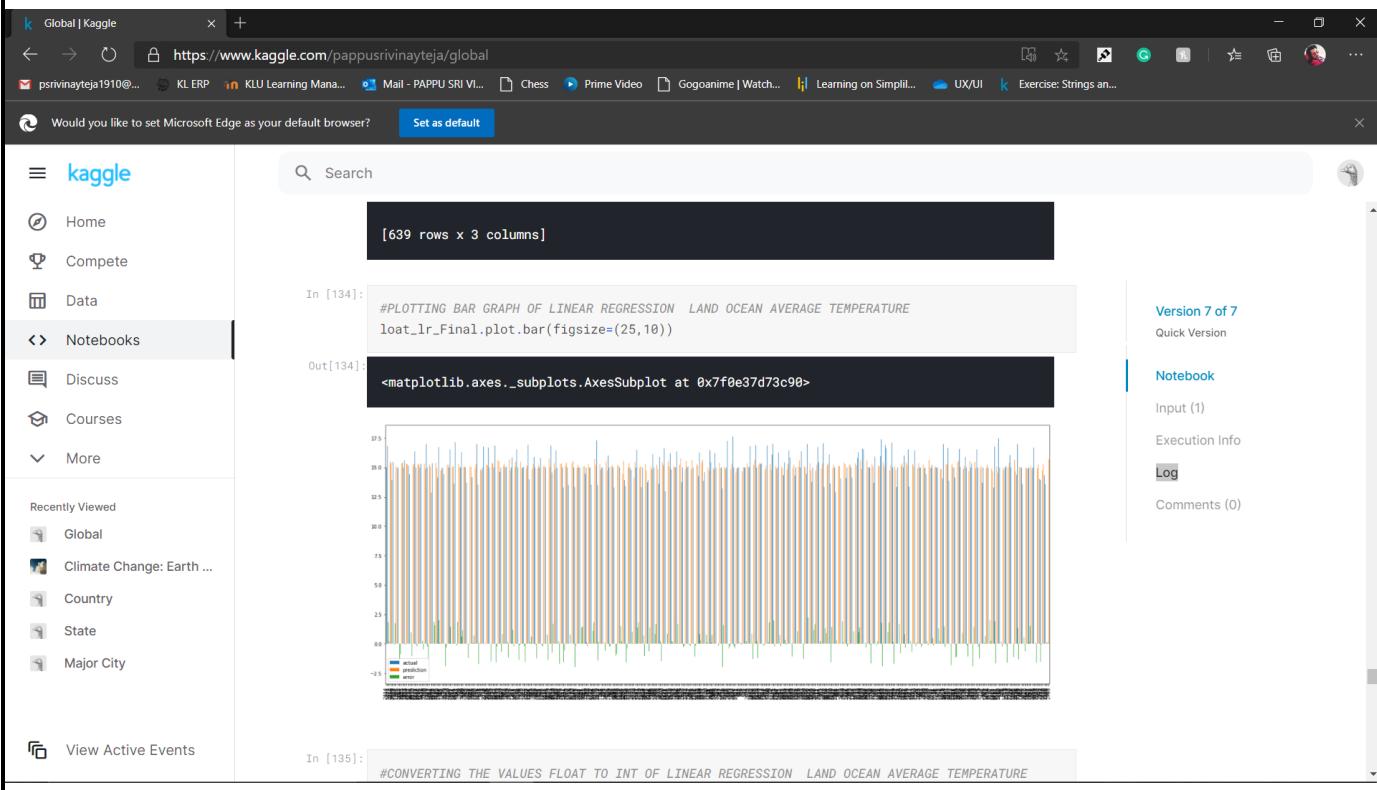


Fig 5.6: Bar Graph of Linear Regression's Land and Ocean Temperature (Global Input)

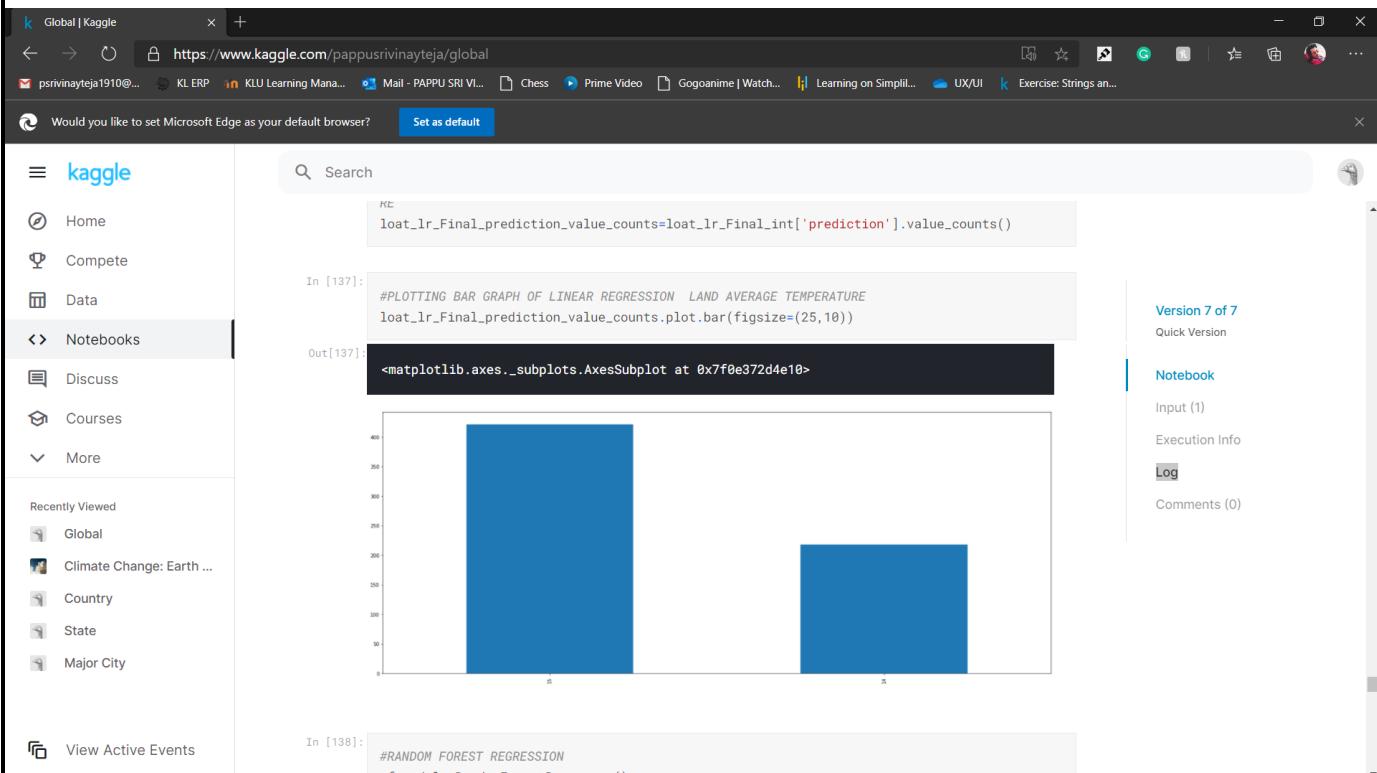


Fig 5.7: Bar Graph of highest count of Linear Regression's Land and Ocean Temperature (Global Input)

Global | Kaggle

https://www.kaggle.com/pappusirivinayteja/global

Would you like to set Microsoft Edge as your default browser? Set as default

**kaggle**

Search

RANDOM FOREST REGRESSION

```

rf_model= RandomForestRegressor()

# fit your model
rf_model.fit(train_X, train_y)

# Calculate the mean absolute error of your Random Forest model on the validation data
loat_rf_prediction = rf_model.predict(test_X)

loat_rf_Final=pd.DataFrame({'actual':test_y,'prediction' : loat_rf_prediction,'erroe':(test_y-loat_rf_prediction)})

print(loat_rf_Final)

```

	actual	prediction	erroe
2414	14.278	13.75755	5.204580e-01
333	15.085	15.00500	2.389264e-14
1471	16.822	16.07481	7.471980e-01
672	15.085	15.00500	1.953993e-14
3113	17.268	15.74098	1.519820e+00
...	...	...	...
1558	14.348	15.09870	-7.587000e-01
2340	13.586	14.22780	-6.418000e-01
459	15.085	15.00500	2.842171e-14
889	15.085	15.00500	1.953993e-14
2795	14.068	16.04866	-1.988660e+00

Version 7 of 7  
Quick Version

Notebook  
Input (1)  
Execution Info  
Log  
Comments (0)

Fig 5.8: Random Forest Regression of Land and Ocean Temperature (Global Input)

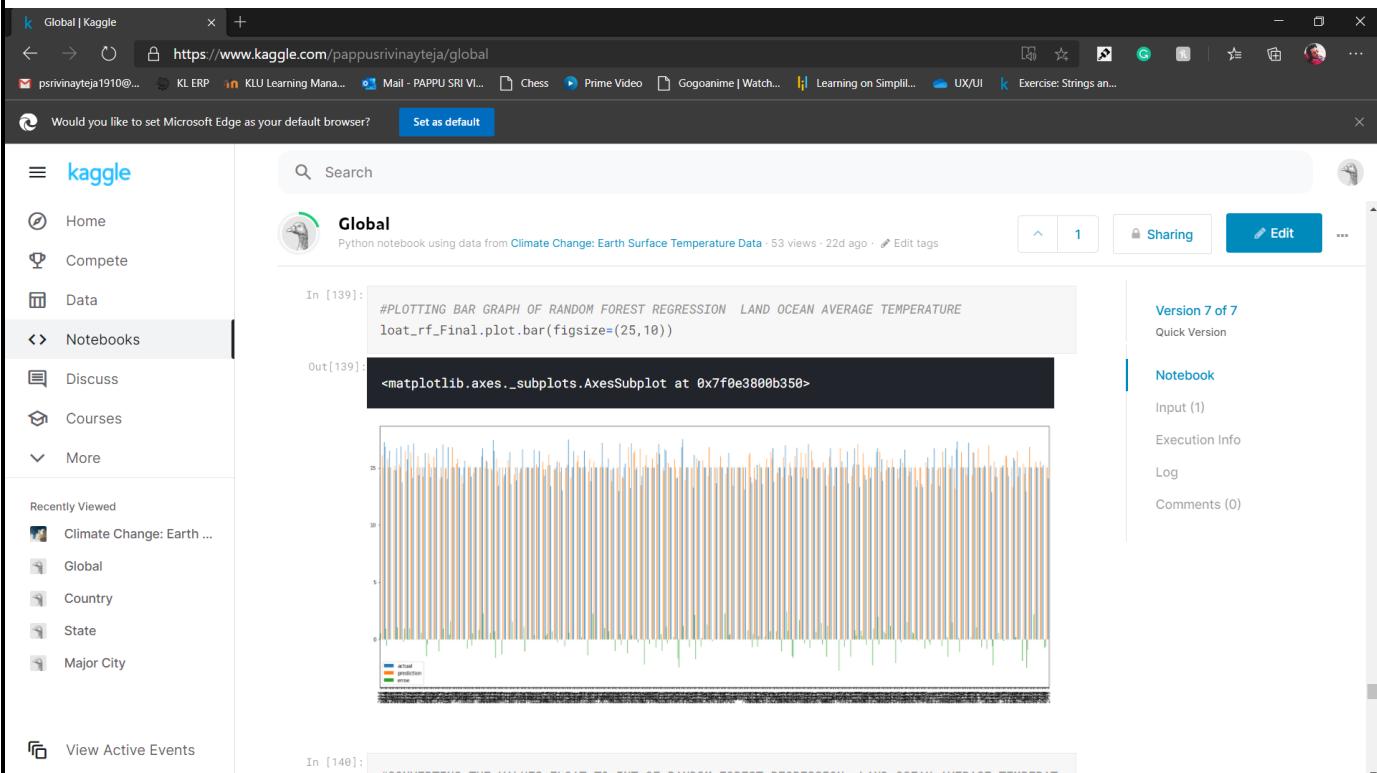


Fig 5.9: Bar Graph of Random Forest's Land and Ocean Temperature (Global Input)

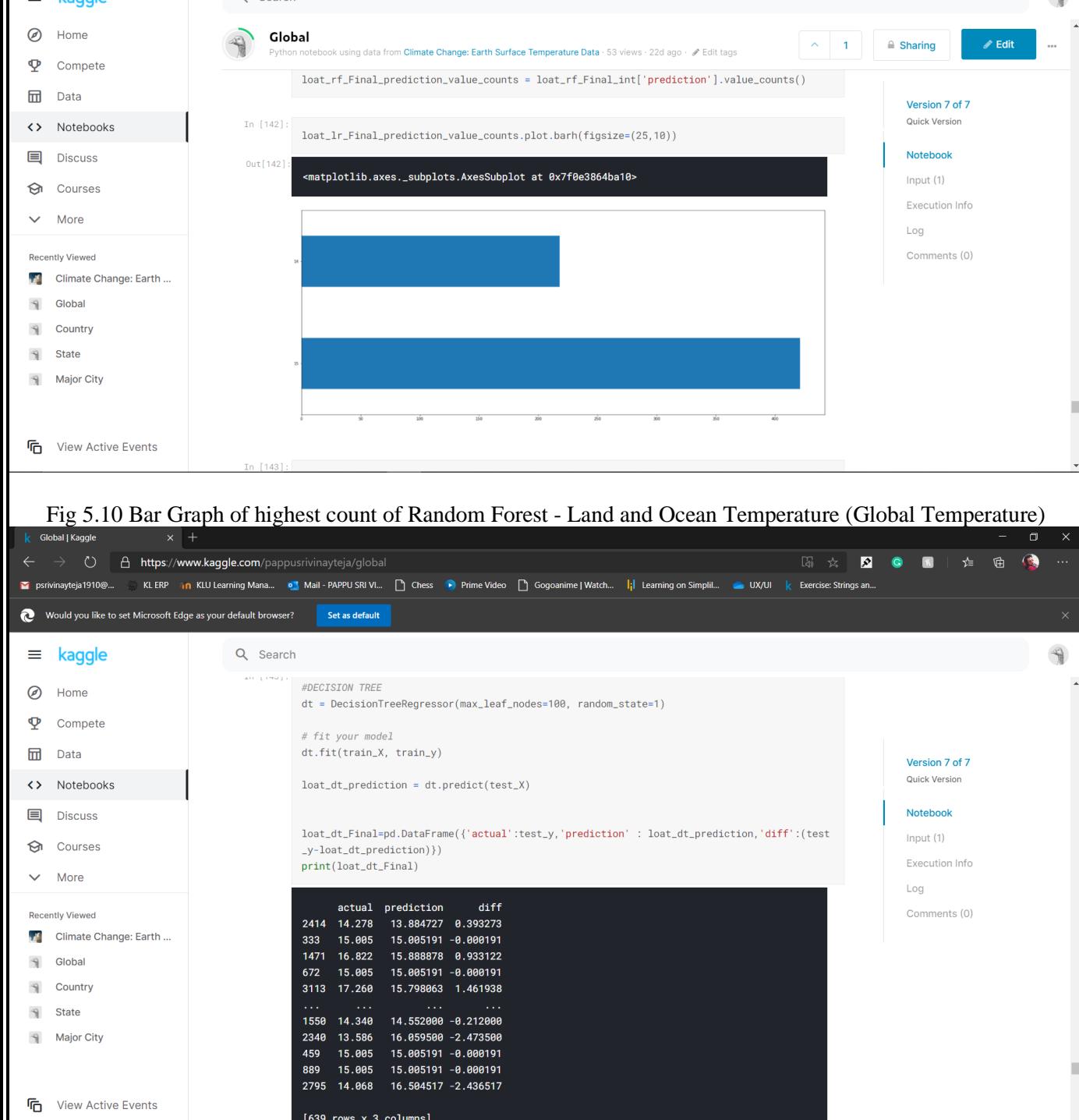


Fig 5.10 Bar Graph of highest count of Random Forest - Land and Ocean Temperature (Global Temperature)

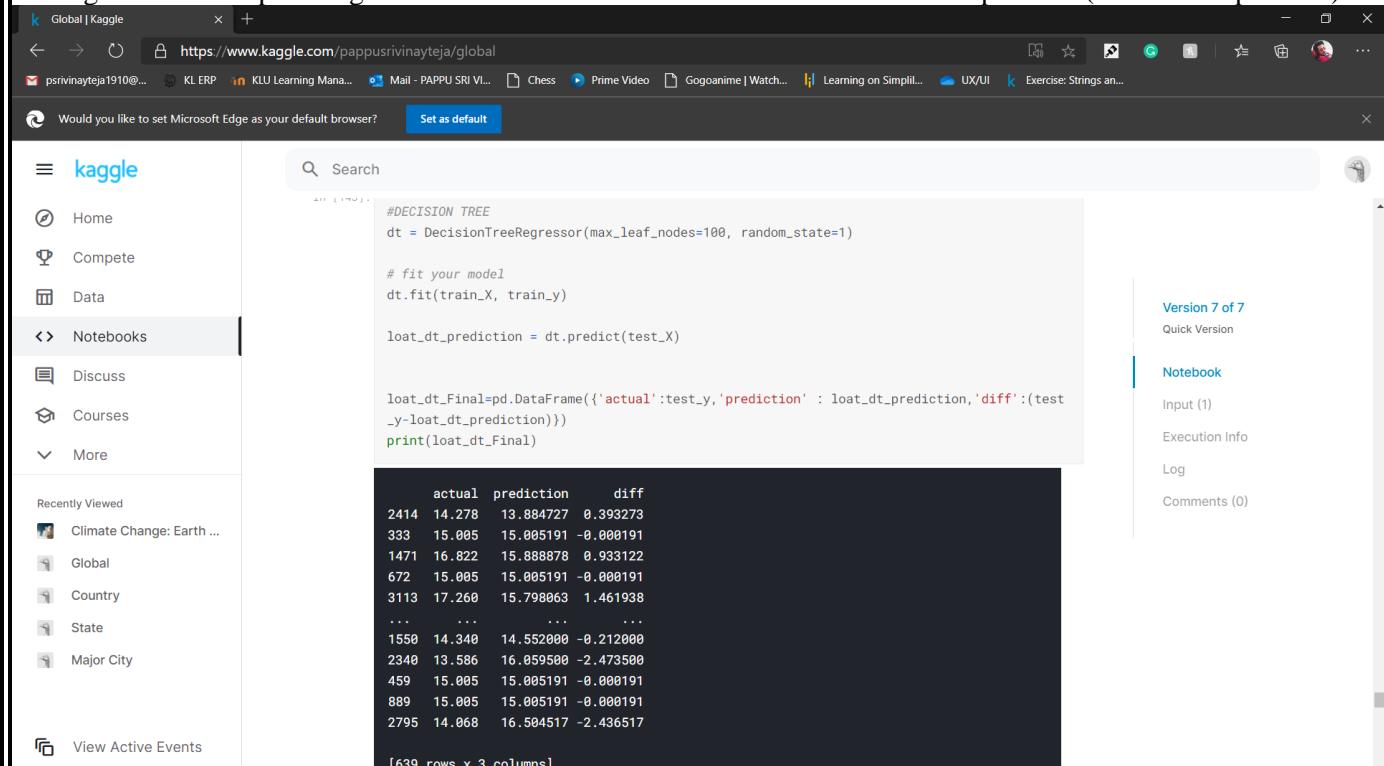


Fig 5.11: Decision Tree of Land and Ocean Temperature (Global Input)

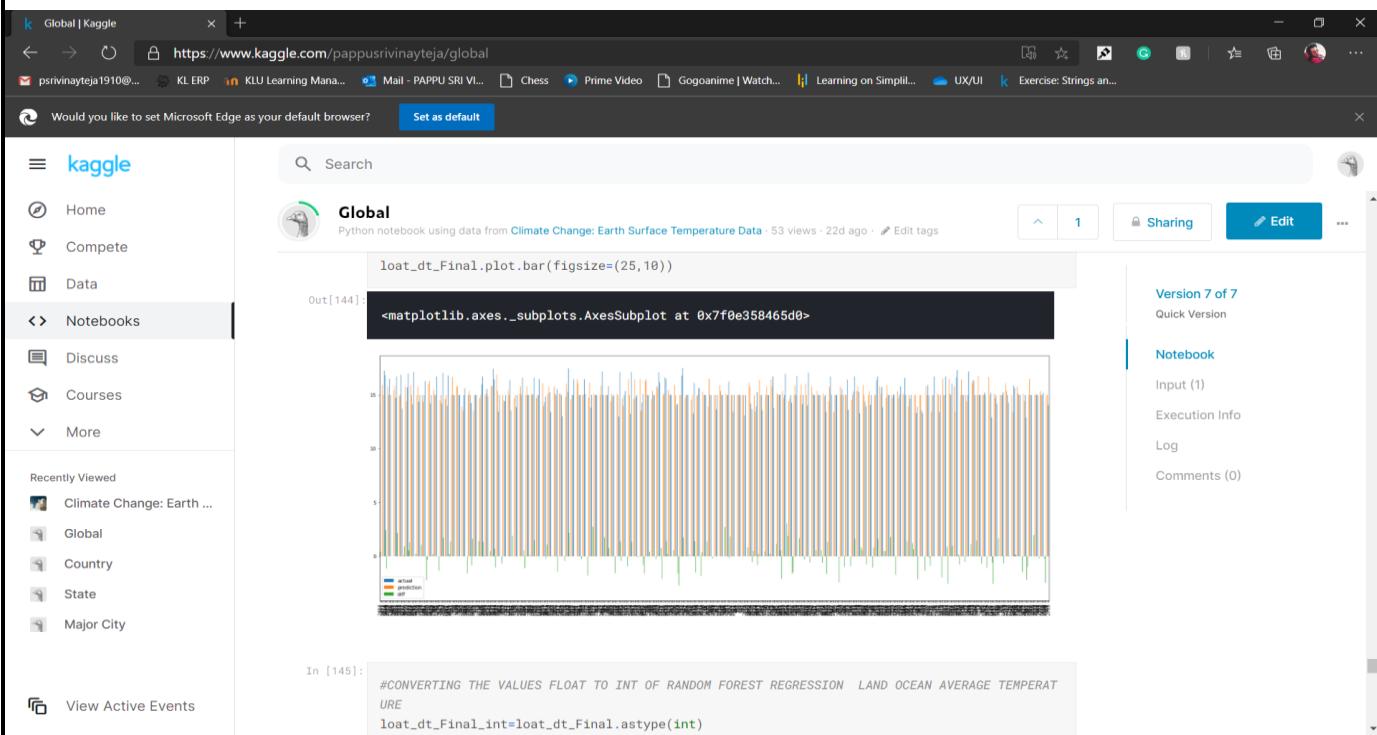


Fig 5.12: Bar Graph of Decision Tree's Land and Ocean Temperature (Global Input)

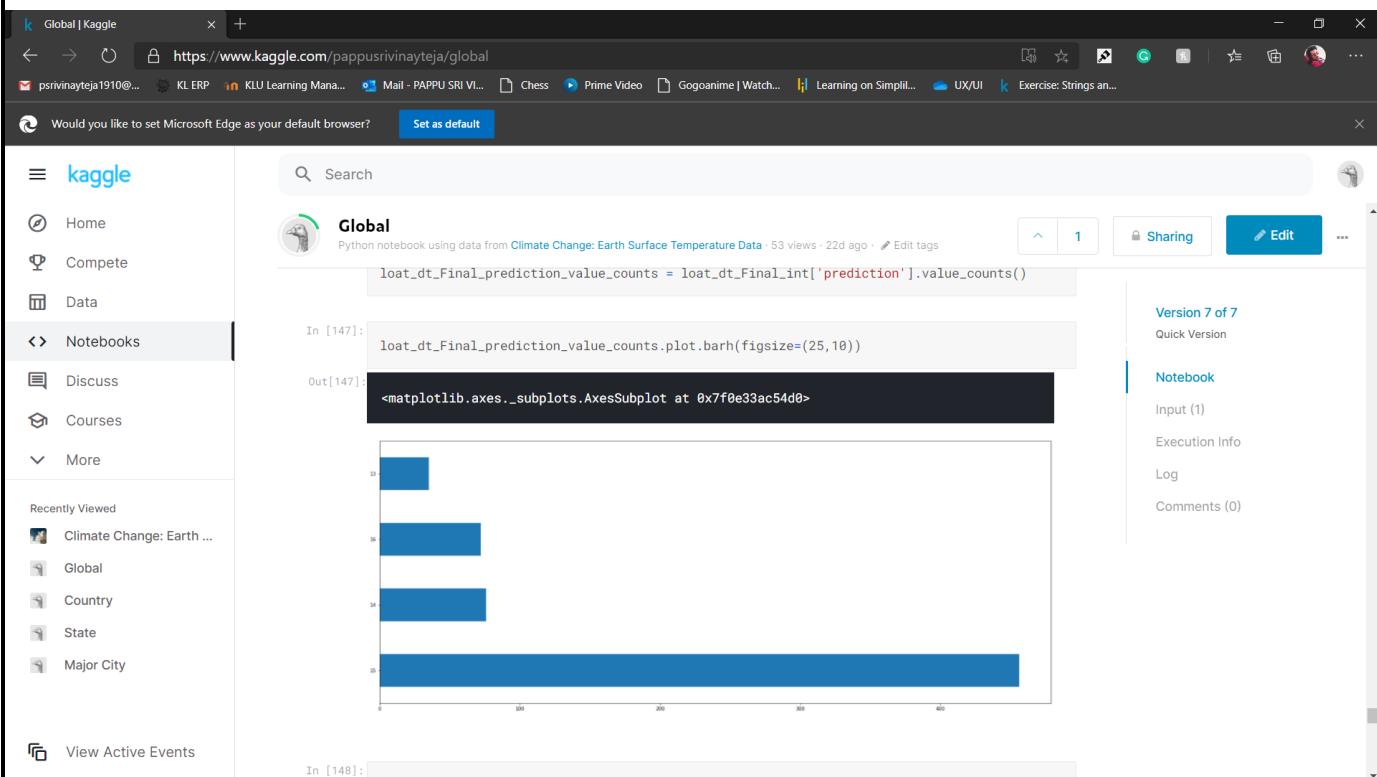


Fig 5.13: Bar Graph of highest count of Decision Tree's Land and Ocean Temperature (Global Input)

Would you like to set Microsoft Edge as your default browser? Set as default

kaggle

Home Compete Data Notebooks Discuss Courses More

Recently Viewed Climate Change: Earth ... Global Country State Major City

View Active Events

Search

Python notebook using data from Climate Change: Earth Surface Temperature Data · 53 views · 22d ago · Edit tags

In [152]:

```
for i in range(0, len(x)):
    #calculating net input for hidden layer z1
    z1=x[i]*w1+x[i]*w3+b1
    #calculating net input for hidden layer z2
    z2=x[i]*w2+x[i]*w4+b2
    outz1=1/(1+math.exp(-z1))
    outz2=1/(1+math.exp(-z2))
    t_out=outz1*w5+outz2*w6+b3
    outy=1/(1+math.exp(-t_out))
    total_error=(1/2)*(y[i]-outy)**2
print('the output of z1 in the hidden layer is',outz1)
print('the output of z2 in the hidden layer is',outz2)
print('net output of y is',outy)
print('the total error is',total_error)
print('net output of y is',outy)
```

the output of z1 in the hidden layer is 0.9975509425953525  
the output of z2 in the hidden layer is 0.9975509425953525  
net output of y is 0.6222290885668514  
the total error is 79.66033358213157  
net output of y is 0.6222290885668514

Version 7 of 7 Quick Version

Notebook

Input (1) Execution Info Log Comments (0)

Fig 5.14: Feed Forward Neural Networks of Land and Ocean Temperature (Global Input)

## **IX. CONCLUSION**

According to the records the global average temperature is  $13.8^{\circ}\text{C}$  before 1700 and the global average temperature is  $14.8^{\circ}\text{C}$  in 2018.

When we get the predicted data, the value is nearly  $15^{\circ}\text{C} - 16^{\circ}\text{C}$ .

I conclude that the temperature was increasing it leads to danger to life on earth.

I also conclude that the Greenhouse effect and its effect theory given by “Irish physicist John Tyndall” is true.

## **X. FUTURE SCOPE**

1. We can develop as complete temperature prediction.
2. We can also develop as weather forecasting application.

## XI. REFERENCE

- [1] S. Karthick, D. Malathi, C. Arun, WEATHER PREDICTION ANALYSIS USING RANDOM FOREST ALGORITHM, Volume 118, No. 20, Special Issue, 2018 ISSN: 1314-3395
- [2] Siddharth S. Bhatkande, Roopa G. Hubballi, Weather Prediction Based on Decision Tree Algorithm Using Data Mining Techniques, Vol. 5, Issue 5, May 2016
- [3] Lakshmi Devasena C, Comparative Analysis of Random Forest, REP Tree and J48 Classifier for Credit Risk Prediction, International Journal of Computer Applications (0975 – 8887).
- [4] Jehad Ali, Rehanullah Khan, Nasir Ahmad, Imran Maqsood, Random Forests and Decision Trees, International Journal of Computer Science Issues(IJCSI), Vol. 9, Issue 5, No 3, September 2012, ISSN (Online) 1694-0814.
- [5] Ron Holmes, “Using a decision tree and neural net to identify severe weather radar characteristics”
- [6] MURAT H. SAZLI, A BRIEF REVIEW OF FEED-FORWARD NEURAL NETWORKS, V.50(1) pp 11-17 (2006)
- [7] M. H. Sazli, C. Isik. “Neural Network Implementation of the BCJR Algorithm”. accepted for publication in Digital Signal Processing Journal, Elsevier, 2005.
- [8] Tianyi Zhao, Jiaming Wang, Meng Xu & Kuishan Li, An online predictive control method with the temperature based multivariable linear regression model for a typical chiller plant system, 13, pages335–348(2020)
- [9] Montgomery DC, Peck EA, Winning GG(2012). Introduction to Linear Regression Analysis, 5<sup>th</sup> ed. Hoboken, NJ, USA: John Wiley & Sons.
- [10] Chan T-S, Chang Y-C, Huang J-H (2017). Application of artificial neural network and genetic algorithm for optimization of load distribution for multi-type-chiller plant. Building Simulation, 10: 711-722.
- [11] Chang Y (2007). Sequencing of chillers by estimating chiller power consumption using artificial neural networks. *Building and Environment*, 42: 180–188.

- [12] Fan C, Ding Y (2019). Cooling load prediction and optimal operation of HVAC systems using a multiple nonlinear regression model. *Energy and Buildings*, 197: 7–17.
- [13] Montgomery DC, Peck EA, Vining GG (2012). Introduction to Linear Regression Analysis, 5th edn. Hoboken, NJ, USA: John Wiley & Sons.
- [14] Goudarzi N, Shahsavani D, Emadi-Gandaghi F, Chamjangali MA (2014). Application of random forests method to predict the retention indices of some polycyclic aromatic hydrocarbons. *Journal of Chromatography A*, 1333: 25–31.
- [15] Yu FW, Ho WT, Chan KT, Sit RKY (2017). Critique of operating variables importance on chiller energy performance using random forest. *Energy and Buildings*, 139: 653–664.