International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 01 | Jan 2021 www.irjet.net p-ISSN: 2395-0072

CLIMATE CHANGE PREDICTION

Mohammed Abdus Samee[1], M. S. Harsha Vardhan Reddy[2], Karthik Suresh[3]

[1,2,3]*Student, Department of Computer Science and Engineering, Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad-500090, Telangana, India*

Abstract - *This project proposes a prediction model to predict the temperature at a specific time in the future of the specified region. This is done by analyzing past temperature records and using Regression Algorithms to predict the temperature. In this project, we will be using Flask Web development to create a user-friendly web application to read the inputs from the user. We will be using certain python libraries such as NumPy, Pandas, Matplotlib, etc. to analyze and extract data from the dataset and to represent the acquired data in form of a graph. According to the output produced, we will be concluding whether the area is influenced by climate change or not.*

Key Words: **Data Analysis, Climate Change, Regional Temperature, Regression Algorithms.**

1.INTRODUCTION

Climate change is one of the major environmental challenges faced by the world today. Climate change is linked with diverse and undesirable impacts on farming, water resources, forest and biodiversity, ocean level increase, and raise in temperature. Reduction in agricultural output is the main impact of climate change. All mainstream inhabitants are dependent on farming in one way or another. Climate change would put added stress on the environmental and socio-economic systems. As such the world is already facing terrific pressure due to speedy industrialization, urbanization, and fiscal development. This paper examines the impact of climate change and its various aspects.

1.1 DOMAIN DESCRIPTION

In this project we will be using Flask Web development to create a user-friendly web application to read the inputs from the users, we will be using certain python libraries such as NumPy, Pandas, Matplotlib, etc. to analyze and extract data from the dataset and to represent the acquired data in form of a graph. This project proposes a prediction model to predict the temperature at a specific time in the future of the specified region. This is achieved by analyzing the past temperature of records using three types of algorithms that are given below.

1. Linear Regression: A linear approach to model the link between a scalar response and one or additional descriptive (independent) variables is called linear regression.
2. Auto Regression: A time series model that uses observations from preceding time steps as input to a regression equation to foresee the value at the next time step. Is called Auto Regression. It is a very easy idea that can result in precise forecasts on a variety of time series problems.
3. The Auto Regressive Integrated Moving Average (ARIMA): It is a class of models that 'explains' a given time series based on its precedent values, i.e., its lags and the lagged forecast error, so that equation can be used to forecast prospect values.

1.2 APPLICATION

We use this to predict the future temperature variations of a particular location of a particular year. This temperature variation prediction will help in contemporary weather forecasting, serious weather alerts, and advisories, which are issued by weather forecasting services in the case that severe or hazardous weather is expected. This is done to guard life and belongings. Some of the most frequently known severe weather advisories are the harsh thunderstorm and tornado warnings, as well as the warnings for the areas that are prone to flood. The study on temperature variations can be used to predict winter weather, extreme winds, floods, tropical cyclones, and fog. Severe weather advisories and alerts are put out through the media, as well as radio, using emergency systems as the Emergency Alert System. These predictions will help in various sectors of the economy. A few of the sectors which will be benefited are given below

1. Agriculture: Farmers depend on weather predictions to decide on what work to do on a particular day. Temperature differences are very significant in deciding what crop to grow.
2. Marine: Business related and recreational use of waterways can be restricted significantly by wind course and speed, wave recurrence and heights, tides, and rainfall. These factors are influenced by temperature variations.
3. Air traffic: The aviation business is very sensitive to the weather; precise weather forecasting is essential bearing in mind the fact that the greatest number of plane crashes recorded all over the world have weather-related causes. Pilots are briefed preceding takeoff on the conditions to anticipate on the route and at their destination. Additionally, airports frequently change which runway is being used to take advantage of a headwind while takeoff and landing.

2. SYSTEM ANALYSIS

2.1 OBJECTIVES

- **To visualize the temperature details of various regions.**

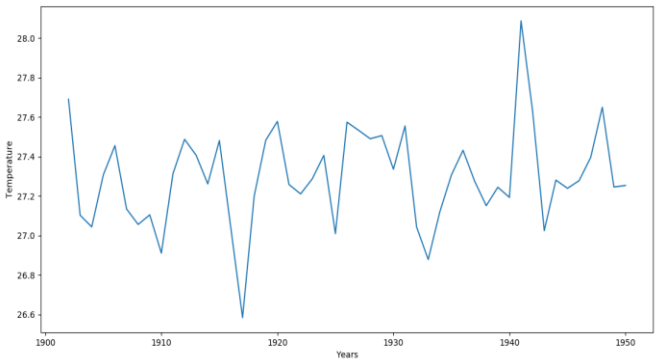


Fig -1: Example of Temperature Representation

- **To Predict the temperature for the given year.**

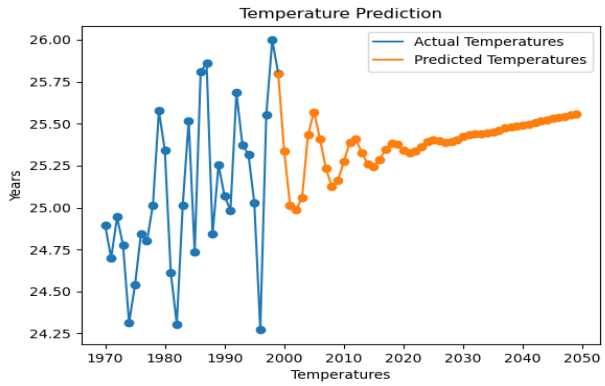


Fig -2: Example of Temperature Prediction Representation

2.2 PROBLEM STATEMENT

- Predicting whether the specified place is affected by climate change or not.

2.3 SOFTWARE/HARDWARE REQUIREMENTS

- OS: Windows 7 or above
- Python IDLE, Python Virtual Environment
- Python Libraries
- Web Browser

3. IMPLEMENTATION

3.1 PROGRAMMING LANGUAGE USED

The programming language which we used to develop this project in Python. Python's main handy features are easy to write and easy to read mannerism, coding in python should reduce one's efforts to develop a project significantly. Since importing and implementing ML libraries is easy in python, we opted for this language to develop this project.

3.2 FRAMEWORK USED

In this project, python's Flask micro-framework is used. "Micro" in micro-framework does not mean that your whole web application has to fit into a single Python file (although it certainly can), nor does it mean that Flask is lacking in functionality. The "micro" in micro-framework means Flask aims to keep the core simple but extensible. Flask won't make many decisions for you, such as what database to use. Those decisions that it does make, such as what templating engine to use, are easy to change. Everything else is up to you so that Flask can be everything you need and nothing you don't. By default, Flask does not include a database abstraction layer, form validation, or anything else where different libraries already exist that can handle that. Instead, Flask supports extensions to add such functionality to your application as if it was implemented in Flask itself. Numerous extensions provide database integration, form validation and upload handling, various open authentication technologies, and more. Flask may be "micro", but it's ready for production use on a variety of needs.

3.3 ALGORITHMS USED

We used multiple regression techniques to predict the temperature using Machine Learning algorithms such as

1) **Linear Regression**
2) **Auto Regression**
3) **Auto Regression Integrated Moving Averages.**

3.3.1 LINEAR REGRESSION

Linear regression is a commonly used predictive analysis model. This module highlights the use of Python linear regression which finds the line of best fit and the coefficient of x. Linear Regression is a predictive modeling technique. It is used whenever there is a linear relationship between the dependent and the independent variables.

3.3.2 AUTO REGRESSION

Auto-regression is a time series model that uses observations from previous time steps as input to a regression equation to predict the value at the next time step. A regression model, such as linear regression, models an output value based on a linear combination of input values. For example: Y = b0 + b1*X. Where Y is the prediction, b0 and b1 are coefficients found by optimizing the model on training data, and X is an input value. This technique can be used on time series where input variables are taken as observations at previous time steps, called lag variables. For example, we can predict the value for the next time step (t+1) given the observations at the last two-time steps (t-1 and t-2). As a regression model, this would look as follows: X(t+1) = b0 + b1*X(t-1) + b2*X(t-2). As the regression model uses data from the same input variable at previous time steps, it is referred to as an auto-regression (regression of self).

3.3.3 AUTO REGRESSIVE INTEGRATED MOVING AVERAGE ALGORITHM (ARIMA)

ARIMA stands for Auto Regressive Integrated Moving Averages, as the name suggests this regression model is designed by combining two other regression models: Auto Regression and Moving Averages with a touch of integration. Autoregressive integrated moving average (ARIMA) models were popularized by Box and Jenkins (1970). An ARIMA model describes a univariate time series as a combination of autoregressive (AR) and moving average (MA) lags which capture the autocorrelation within the time series. The order of integration denotes how many times the series has been differenced to obtain a stationary series. We write an ARIMA (p, d, q) model as:

$$\Delta^D y_t = \sum_{i=1}^{p} \phi_i \Delta^D y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t$$

$$\epsilon_t \sim N\left(0, \sigma^2\right)$$

ARIMA models are associated with a Box-Jenkins approach to time series. According to this approach, you should difference the series until it is stationary, and then use information criteria and autocorrelation plots to choose the appropriate lag order for an ARIMA process. You then apply inference to obtain latent variable estimates and check the model to see whether the model has captured the autocorrelation in the time series. For example, you can plot the autocorrelation of the model residuals. Once you are happy, you can use the model for retrospection and forecasting. The ARIMA model is a class of statistical models for analyzing and forecasting time series data. It explicitly caters to a suite of standard structures in time-series data, and as such provides a simple yet powerful method for making skillful time-series forecasts. A linear regression model is constructed including the specified number and type of terms, and the data is prepared by a degree of differencing to make it stationary, i.e., to remove trend and seasonal structures that negatively affect the regression model. A value of 0 can be used for a parameter, which indicates to not use that element of the model. This way, the ARIMA model can be configured to perform the function of an ARMA model, and even a simple AR, I, or MA model. Adopting an ARIMA model for a time series assumes that the underlying process that generated the observations is an ARIMA process. This may seem obvious but helps to motivate the need to confirm the assumptions of the model in the raw observations and the residual errors of forecasts from the model.

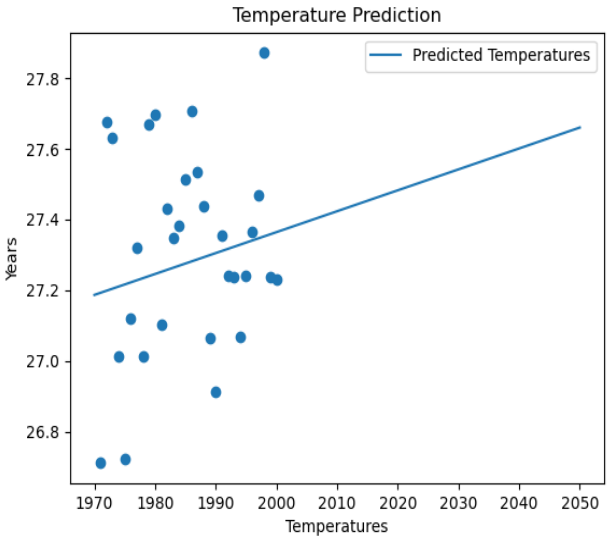
4. SAMPLE CODE AND OUTPUTS

4.1 LINEAR REGRESSION

```
def linear_regression(full_data, value):
    if os.path.exists("F:/Python/FullMiniProject/static/figure.png"):
        os.remove("F:/Python/FullMiniProject/static/figure.png")
    dates = full_data[['Date']]
    temperatures = full_data['Temp']
    lin_reg = LinearRegression()
    lin_reg.fit(dates, temperatures)
    result = lin_reg.predict([[value]])

    years = range(1970, 2050 + 1)
    values = []
    for i in range(1970, 2050 + 1):
        values.append([i])
    predictions = lin_reg.predict(values)
    plt.plot(full_data['Date'][70:], full_data['Temp'][70:], label="Actual Temperatures")
    plt.plot(years, predictions, label="Predicted Temperatures")
    plt.scatter(full_data['Date'][70:], full_data['Temp'][70:])
    plt.xlabel('Temperatures')
    plt.ylabel('Years')
    plt.title('Temperature Prediction')
    plt.legend()
    plt.savefig("F:/Python/FullMiniProject/static/figure.png")
    return result
```

Output:



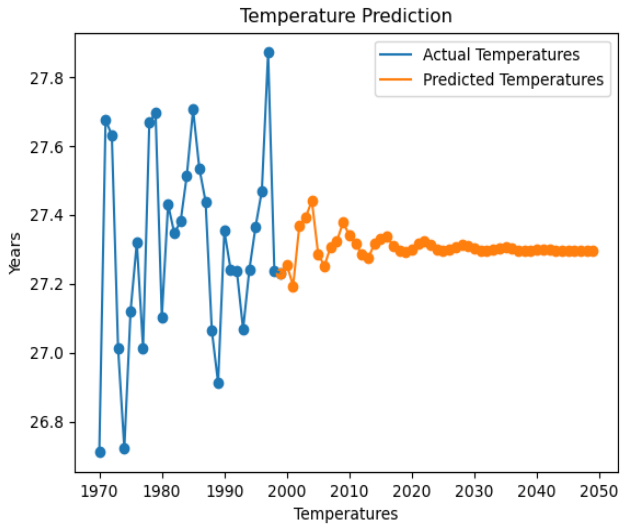
4.2 AUTO REGRESSION

```
def AutoRegression(series, value):
    if os.path.exists("F:/Python/FullMiniProject/static/figure.png"):
        os.remove("F:/Python/FullMiniProject/static/figure.png")
    X = series.values
    Xlen = len(X)
    for i in range(2001, value + 1):
        model = AR(X)
        model_fit = model.fit()
        y = model_fit.predict(len(X), len(X))
        X = list(X)
        X.append(y)
        X = np.asarray(X)
    result = X[len(X) - 1]
    plt.plot(list(range(1970, 2000)), X[70:Xlen], label="Actual Temperatures"
    plt.scatter(list(range(1970, 2000)), X[70:Xlen])

    if value <= 2050:
        for i in range(value+1, 2050+1):
            model = AR(X)
            model_fit = model.fit()
            y = model_fit.predict(len(X), len(X))
            X = list(X)
            X.append(y)
            X = np.asarray(X)

    years = range(1999, 2050)
    plt.plot(years, X[99:150], label="Predicted Temperatures")
    plt.scatter(years, X[99:150])
    plt.xlabel('Temperatures')
    plt.ylabel('Years')
    plt.title('Temperature Prediction')
    plt.legend()
    plt.savefig("F:/Python/FullMiniProject/static/figure.png")
    return result
```

Output:



4.3 ARIMA

```
def difference(dataset, interval=1):
    diff = list()
    for i in range(interval, len(dataset)):
        value = dataset[i] - dataset[i - interval]
        diff.append(value)
    return np.array(diff)


def inverse_difference(history, i, interval=1):
    return i + history[-interval]
```

```
def evaluate_arima_model(X, arima_order):
    train_size = int(len(X) * 0.66)
    train, test = X[0:train_size], X[train_size:]
    history = [x for x in train]
    predictions = list()
    for t in range(len(test)):
        model = ARIMA(history, order=arima_order)
        model_fit = model.fit(disp=0)
        yhat = model_fit.forecast()[0]
        predictions.append(yhat)
        history.append(test[t])
    error = mean_squared_error(test, predictions)
    return error


def evaluate_order(dataset):
    dataset = dataset.astype('float32')
    p_values = range(0, 12)
    d_values = range(0, 5)
    q_values = range(0, 12)
    best_score, best_cfg = float("inf"), (7, 0, 1)
    for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p, d, q)
                try:
                    mse = evaluate_arima_model(dataset, order)
                    if mse < best_score:
                        best_score, best_cfg = mse, order
                except:
                    continue
    return best_cfg
```

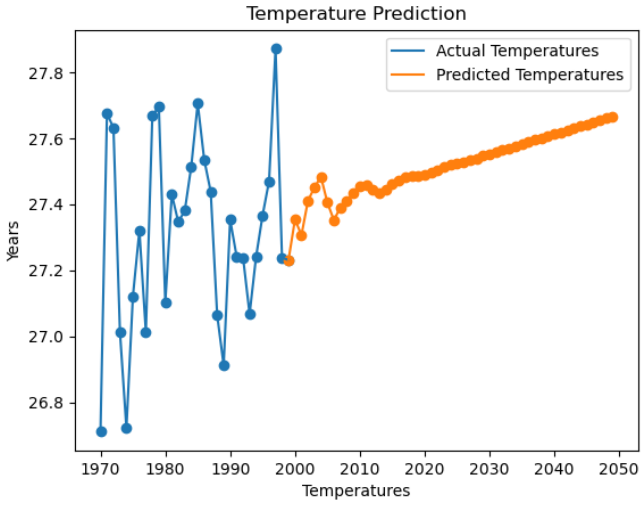
```
def ARIMA_model(full_data, year):
    if os.path.exists("F:Python/FullMiniProject/static/figure.png"):
        os.remove("F:/Python/FullMiniProject/static/figure.png")
    x = full_data.Temp.values
    dif = difference(x, 1)

    arima_order = evaluate_order(full_data)
    model = ARIMA(dif, order=arima_order)
    model_fit = model.fit(disp=0)
    if year <= 2050:
        forecast = model_fit.forecast(steps=50)[0]
    else:
        forecast = model_fit.forecast(steps=year-2000)[0]
    history = [i for i in x]
    for i in forecast:
        value = inverse_difference(history, i, 1)
        history.append(value)

    plt.plot(list(range(1970, 2000)), full_data['Temp'][70:], label="Actual Temperatures")
    plt.scatter(list(range(1970, 2000)), full_data['Temp'][70:])
    years = range(1999, 2050)
    plt.plot(years, history[99:150], label="Predicted Temperatures")
    plt.scatter(years, history[99:150])
    plt.xlabel('Temperatures')
    plt.ylabel('Years')
    plt.title('Temperature Prediction')
    plt.legend()
    plt.savefig("F:/Python/FullMiniProject/static/figure2.png")

    return history[year-1900-1]
```

Output:



5. CONCLUSIONS

A temperature prediction web application developed by using Python and Machine Learning algorithms has two functions:

1. Produces graphical representation of past recorded temperature variation in the range given by the user between 1901 and 2011.
2. Predicts the temperature of a place given by the user.

The project begins by selecting a place and the year for predicting the average temperature of that particular year of the selected location using the past data sets. We also use these data sets to create a graph of the temperature variation of the selected location for the past years (1901-2011). We use three different regression algorithms to predict the future temperature, out of which, the ARIMA model gives the most accurate result.

This project can be enhanced by using more complex algorithms and using more atmospheric factors. It can be trained dynamically as the new temperature data arrives to improve the prediction. When sudden changes in the temperature trends are observed then a climate change can be identified.

REFERENCES

[1] https://www.kaggle.com/
[2] https://www.tutorialspoint.com/flask/index.htm
[3] https://machinelearningmastery.com/autoregression-models-time-series-forecasting-python/
[4] https://towardsdatascience.com/a-beginners-guide-to-linear-regression-in-python-with-scikit-learn-83a8f7ae2b4f?gi=c653fdbec213
[5] https://machinelearningmastery.com/make-manual-predictions-arima-models-python/
[6] https://www.statsmodels.org/stable/generated/statsmodels.tsa.arima_model.ARIMA.html