**Cloud Assignment 2 - Weather Forecasting**

**Live Endpoint:** [**http://ec2-34-243-220-60.eu-west-1.compute.amazonaws.com/**](http://ec2-34-243-220-60.eu-west-1.compute.amazonaws.com/)

**Youtube:** [**https://youtu.be/s-dS01yyPXY**](https://youtu.be/s-dS01yyPXY)

**Introduction**

The purpose of this application is to provide users with an insight into historic and forecasted weather of Delhi. When looking at historic data, users can choose a date from a calendar. They are presented with the temperature of the selected date. Users are given an in-depth look at weather forecasting, with the ability to view the expected forecasts on a monthly, weekly, daily and hourly basis through an interactive graph. This app aims to help users prepare for the day ahead by providing accurate weather predictions. The motivation for this application was to create a useful and reliable tool for Delhi locals and tourists. The final implemented product will take input from users and provides a prediction of weather and the user can see the data visualization of forecasted data through a click of the button.

**Choice of technologies**

*Cleaning and Preprocessing:* Apache Spark - PySpark (ETL Process)

*Web page:* HTML, CSS, Bootstrap, JQuery

*Model creation and Visualisation:* Numpy, Pandas, Seaborn, Plotly offline, Statsmodels

*Integration:*Flask framework*Coding language***:** Python

Cloud platform: AWS - EC2

**Preprocessing Data querying and Storing**

Dataset chosen is a large public data available from Kaggle (shorturl.at/gtvxK), The dataset we used was a Delhi weather dataset for our application, Preprocessing of the data is done using Pyspark. PySpark is a python shell for spark, released by Apache Spark community to support python with Spark. Using PySpark, the user can easily integrate and work with RDD in a python programming language, initially the required columns for our data processing were noted and later on, respective calculations were done on them depending on the requirement, Schema was created and then the data was imported from CSV using the read() function. the average was taken for these for pressure, dewpoint, temperature, and wind-direction columns, this was done by using the mean() function, at first the average or mean value was calculated and then stored in a variable later on using the fill function, we appended the mean values to all the cells which had null values.

But for humidity, the values were far spread so it was apt to ignore all the null values and therefore removing the entire row, this we did with the use of the drop() function, the parameter subset for this was humidity, so that means all the rows which had no values for humidity were discarded. And finally, the resulting dataset was exported from the Jupyter Notebook using the write() function.

The Output of the Jupyter Notebook is given for data exploration and further analysis and is given as input to the prediction model model that was created in python language. The model uses ARIMA model to give the best fit for forcasting. ARIMA (AutoRegressive Integrated Moving Average) is a class of model that captures a suite of different standard temporal structures in time series data. We divide analysed dataset into train dataset and test dataset and feed into the model with time as index and only field temperature which is further used for prediction. The output of the data is stored in CSV format which is then queried using users input on date and displayed on the screen. All the components of the entire application is integrated through flask. The user will enter a particular date of choice into the calender and the prediction model runs in the background generates the csv with date and predicted value and the model queries the csv using users input and provides the user with the temperature for the date given.

**Related Work**AccuWeather and Google weather prediction are well-known weather prediction sites providing accurate weather forecasting to its users.

**Challenges and lessons learned**

Creating the model and integrating the model into the Flask framework was challenging. Deploying the application on the cloud had a weight of its own. We attempted to use Azure and Heroku to host our application but were unsuccessful because Python has dropped support for Windows wheel file pywin32 and we couldn't run the application in Heroku. hence We decided to go with Amazon Web Services cloud platform. While trying to host the application on AWS we faced multiple issues, the major one was getting the configuration and firewall setting right and converting the elastic IP address to a static IP so that the application can be accessed by the outside world. Amazon Web Services sets a default value for the main application.py file. Your application must be set to the same name as the default name or alternatively, the default name must be changed to match your main python file. Otherwise, Elastic Beanstalk will be unable to find your main file. By default, new EC2 instances are blocked from external access. The security groups must be configured to allow access on port 80 in order for the application to be accessible. The application had dependencies on third-party Python packages. It was necessary for us to generate a requirements.txt file in order to allow the EC2 virtual environment to install the required packages.

**Responsibility statement (which group member did what?)**

The whole team was involved in the planning process which consisted of brainstorming, deciding on technologies to be used, and application design.

We have outlined the individual contribution of each team member below.

Student 1 **-** *Satisfactory*

Vaibhav and Gautham carried out data gathering from Kaggle.

Exploration, cleaning and resampling the data according to the required format to be used by the application.

Attempted Microsoft Azure deployment.

Student 2 - *Satisfactory*

Attempted Heroku deployment with Sachin.

Cleaning data with Gautham.

Attempted to deploy application on AWS Elastic Beanstalk.

Student 3 - *Satisfactory*

He carried out data analysis on the cleaned dataset.

From this, he created a data visualization with Python which was one of the key features of the application.

Attempted AWS Beanstalk deployment with Vaibhav.

Student 4 - *Satisfactory*

Web page(UI) development was handled by Bharath.

He successfully deployed the application on the AWS EC2 along with Maureen.

Midway Report generation was done by Bharath and Meghana.

Student 5 - *Satisfactory*

Created a repository for Cloud Project in Github and maintenance.

Created the temperature prediction model with Meghana.

Attempted Heroku deployment.

Final report preparation.

Integration of application

Maureen - *Satisfactory*

Successfully deployed the application on AWS with Bharath

Attempted to deploy the application on GCP

Final Report Creation was done by Maureen and Meghana

Integration of the application with Meghana

Student 6 - *Satisfactory*

Integration of the application using flask

Midway report preparation with Bharath

Model creation and Report generation

**Future Work**

To make the application more useful for our users we would like to incorporate a clothing and accessory recommendation feature. The recommendations would be based on the predicted weather for that day. This would further assist our users in their daily planning and activities.