Global Temperature Prediction
CS649 Big Data Tools & Methods

Daniel Barnas
College of Sciences
Computer Science
San Diego State University
San Diego, California 92182

Yousif Jabbo
College of Sciences
Computer Science
San Diego State University
San Diego, California 92182

Steven Thanhtan Nguyen
College of Sciences
Computer Science
San Diego State University
San Diego, California 92182

Abstract—**Climate change is a pressing global issue that is affecting the world's temperature patterns. In this research project, 4 different Time Series Forecasting Models, specifically Facebook Prophet, ARIMA (Autoregressive Integrated Moving Average), SARIMAX (Seasonal ARIMA), and LSTM (Long Short-Term Memory), are used to try to predict the average temperature of a city 2 years into the future. LSTM was not able to be used as not enough computing resources were available. ARIMA was not an optimal model as the data was seasonal. When utilizing SARIMAX for monthly predictions, SARIMAX received an average RMSE of 2.81. When utilizing SARIMAX for weekly predictions, SARIMAX received an average RMSE of 5.17. As for Facebook Prophet, the average RMSE was 5.74. Despite these results we found that Facebook Prophet was the best model in terms of prediction speed and its ability to predict daily temperatures, for which SARIMAX took too long in comparison. An interactive dynamic visualization demonstrating the predictions was then created utilizing D3, a Javascript library for data visualizations, to demonstrate the differences between the models. Linear Regression was also utilized to generate a mathematical equation to predict the change in temperature overtime.**

1. Introduction

Temperature has always been a concern for many people, regardless of where they live. A common concern people have about future temperature is the temperature patterns and their impact on various aspects of life. The prediction of the temperature of a city in the future can provide many benefits to people such as how their crops may flourish, how their electricity consumption may be affected, how they are more susceptible to illnesses, and much more. At the moment, there are many Time Series Forecasting Machine Learning Models, but which model is the best to predict the average temperature of a city in the future?

1.1. Objective

The objective of this research project is to predict the average temperature for 325 cities in the world 2 years into the future. Multiple features are used as inputs into the Machine Learning Models such as Date, Region, Country, State, City, Longitude, Latitude, and Average Temperature. The models will learn how to predict the temperature given some or all of these features. The test of predictions will be on Facebook Prophet, ARIMA, SARIMAX, and LSTM, to see which Machine Learning Model performs the best for predicting a city's future temperature for 2 years.

1.2. Project Structure

In this research project, there are 7 main sections:

- Section 1 will be providing a brief introduction and overview of our project.
- Section 2 will be providing a description of the dataset that was used.
- Section 3 will be providing the problem statement that our project aims to address.
- Section 4 will describe the methodology that was used to solve the problem.
- Section 5 will explain and describe our results.
- Section 6 will discuss and interpret our results.
- Section 7 will be the conclusion of our research project in which we summarize our main findings, explain how they contribute to the domain of Big Data, and talk of potential future improvements.

2. Dataset Description

This section will be discussing the raw dataset used for this project as well as how the data was cleaned for processing.

2.1. Raw Data & Source

The raw dataset utilized for this project was obtained from Kaggle called ”Daily Temperature of Major Cities”. The name of the file is ”city_temperature.csv”. The raw dataset was 140.6MB in size and had 2906327 records. There is an average of 24 years' worth of daily temperature data available for each city. Within this raw dataset, there were 8 features provided. Those features were Region, Country, State, City, Month, Day, Year, and Average Temperature. Upon observation, the State column of the raw data is seen to contain many nulls. This is due to the fact that not all countries have states. As for the Average Temperature Column, many values are seen to be -99, which are taken to be as filler values. The link of the dataset can be found here https://www.kaggle.com/datasets/sudalairajkumar/daily-temperature-of-major-cities.

2.2. Cleaned Data

4 main actions were taken in order to clean the data before being processed into our Machine Learning Models. The first action was to combine the State column and the City column of the raw dataset. This is because some states had the same city name. The second action was to try to find the longitude and latitude of the cities so that there could be more features to pass into our models. To receive the longitude and latitude of the cities, another dataset was utilized. This dataset was obtained from Kaggle called ”World cities database”. The link for that dataset can be found here https://www.kaggle.com/datasets/juanmah/world-cities. The name of the file is ”worldcities.csv”. This dataset was 4.73MB in size and had 44691 records. Within this dataset, there were 10 features provided. Those features were City, City as an ASCII String, Latitude, Longitude, Country, Alpha-2 Code of the Country, Alpha-3 Code of the Country, Admin Name, Capital, and Population. Some cities were dropped in the worldcities.csv file as not all the cities in that file were in the raw dataset, thus creating the worldcities_modified.csv file. Because we only needed the Latitude and Longitude of the cities in our raw dataset, we decided to merge the raw dataset with the ”worldcities_modifed.csv” file to obtain the Latitude and Longitude of each city. The third action taken was to remove the records in the dataset that had a Average Temperature value of -99 as those were filler values. The fourth action was to convert the Year, Month, and Day columns of the dataset into a single date column to be used as one of the input features for our models. After performing all 4 of these data manipulation tasks, the data was ready for processing into our models.

3. Problem Statement

This section will be stating the problem statement the project aims to address as well as why this issue is important.

3.1. Problem Importance & Relevance

Predicting the future temperature of a city is a significant concern, as it can impact crop yields, electricity usage, public health, and the need for government action to address climate change. However, this task remains challenging since the temperature of a city in the future is uncertain.

3.2. Research Question to Answer

What is the expected temperature trend for a specific city over the next 2 years and what is the rate of temperature change over time? These are the questions that we are attempting to answer.

4. Methodology

In this section, we will be describing the methodology used for completing this project. To solve the problem of forecasting temperatures based on historic data, we used a few different approaches. Generally, time-series forecasting algorithms fall into two categories. There are univariate algorithms such as ARIMA that, as the name suggests, rely on a single variable for their predictions. They observe the value of a parameter over time and attempt to extrapolate that variable into the future. On the other hand, there are multivariate algorithms that can take many inputs when trying to forecast. These include many deep learning approaches such as Long short-term memory (LSTM). These allow you more flexibility in the types of inputs you can feed your model. In our case, additional inputs we wanted to explore included the latitude and longitude of cities. One of our goals was to see if additional inputs would result in multivariate models generating better predictions. To start, we will discuss our methodology of running univariate models on our temperature data.

4.1. Initial Limitations

Most of the initial limitations of our project stem from the sheer number of cities in the data that we want to predict temperatures for. A limitation for some of the cities is that not all cities had the same amount of data. An example of that is that some cities latest data would be in the year 2020, but other cities would have latest data in only years prior to that like 2012. In addition, some cities would have missing days. It’s one thing to run an ARIMA model on a single time-series. But running it on 300+ cities is taxing and time-consuming for any single computer. As for LSTM, the model was not able to be trained to its full potential as this model required more computing resources than were available. Below in section 6, we discuss the models we chose to employ and how we overcame this limitation.

4.2. Algorithms, Techniques, & Tools

The univariate models we chose to implement are SARIMA and Prophet (by Meta). We will begin by discussing each model and how it works when applied to a single datapoint (in our case, a single city). Later, we will explain the steps taken to parallelize the process to run over all the cities in the data on a Hadoop cluster running PySpark.

4.3. SARIMAX

SARIMAX can be thought of as an upgraded form of ARIMA. ARIMA stands for "auto-regressive integrated moving average" and it works by creating a linear equation that extrapolates a forecast based on historical data [1]. It is made of 3 components: Autoregression (AR), Integrated (I), and Moving Average (MA) [2]. An autoregressive model is a model in which the variable depends on past values of itself. Integrated refers to the process in which the time series is made stationary by subtracting prior values from current values (called differencing). This step removes any non-constant trends, allowing for more accurate forecasts. Overall, ARIMA is bread-and-butter time-series forecasting model that is very popular.

However, ARIMA is not without its weaknesses. One such major weakness is that ARIMA is not well-suited for seasonal data. ARIMA works best on "stationary" data which is data in which the mean and variance remain mostly constant over time [3].Our data appears to be seasonal but to confirm we used Augmented Dickey Fuller Test(ADF). The ADF is a statistical test used to test if our data is stationary. Unfortunately, the test showed that our data is clearly seasonal, which means that ARIMA is not a great fit. All hope is not lost, however, because we have SARIMA to save the day. SARIMAX stands for Seasonal-ARIMA and it essentially incorporates seasonality into ARIMA to make it more robust. It does this by incorporating a few additional hyperparameters whereby it can detect trends over the seasonal period that you specify [4].

To run SARIMAX on a single city, we begin by creating a dataframe consisting of the temperatures recorded for all dates for that city. To run, ARIMA needs a few parameters. These are p, also known as the lag order, d, also known as the degree of differencing, and q, or the moving average window [2]. We run the auto-arima method of the pmdarima package, which detects the optimal parameters to run for an ARIMA model. It works by conducting various differencing tests such as Augmented Dickey-Fuller to find the best order of differencing [5].

Knowing the optimal parameters, we created a SARIMAX model based on the temperature column of our data and the expected seasonality period. In the case of our data which consists of daily measurements, the obvious seasonality period is 365 to match the days in a year. However, through experimentation, we found that a seasonality period of 365 causes the model to run for a VERY long time. We found that we were not the first to encounter this issue [6]. Therefore, we decided to run two versions of SARIMAX. The first would predict temperatures by month and the second by week. This approach assumes that the average temperature over a week (best-case) or month (worst-case) does not vary too wildly, and that the temperature on most days will fall near the average. We will later see how well this assumption holds.

To get weekly and monthly predictions, we aggregated the input data by week or month respectively and gathered the average over those intervals to use as the inputs instead. Instead of providing daily predictions, this approach would provide predictions by week or by month instead. While not providing the same granularity as daily predictions, this gave us a way to check how weekly or monthly predictions perform and whether they could be "good enough" given the improvements in processing speed.

Once we split the data into the proper intervals, we established an accuracy measurement timeframe of 2 years, meaning that we would use the predictions of each model over the last 2 years of available data to measure accuracy. We also established a prediction window of 3 years after the last available data. In either the weekly or monthly case, this required creating a new datetime index consisting of a range of dates over the total 5 year window in the proper frequency (weekly or monthly). At this point, the SARIMAX model was ready to be fitted to the training data. With a fitted model, we could then make predictions over the established intervals. This is the method in which SARIMAX was applied to make predictions for a single city. We will discuss how this was parallelized later.

4.4. Prophet

Prophet is an algorithm for time-series forecasting created by Facebook (now Meta). It overcomes many of the issues ARIMA and SARIMAX face when dealing with non-stationary data, or data with various seasonality effects [7]. In fact, it is best-suited for data that demonstrates clear seasonality, and it boasts good results without much in the way of tuning [8].

It is made up of three functions and an error term [9]:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

The first function is the growth function which models the trend of the data. While traditionally, the trend could be obtained by plotting a regression line against the data, Prophet takes this a step further by detecting changes in the trend and marking those as "changepoints." It can then create multiple growth functions for each interval delineated by changepoints. This allows it to overcome the stationarity constraints of other time-series forecasting models. Each growth function can be linear, logistic, or flat.

The second function is the seasonality function, which is a Fourier Series capable of detecting cyclical components in the data and mapping them to sine and cosine functions. This gives Prophet incredible power in detecting any seasonal cycles in the data, no matter the timeframe [7]. The third function is the holiday function, which deducts or adds to a forecast based on historical spikes or dips on a given holiday. Since our data is based on physical temperature measurements, this function does not really impact our predictions so we will not go into much depth. The epsilon term represents any unusual errors that may appear in the prediction.

Prophet works by attempting to fit a curve to the data that contains the three components above. The functions that lead to the best fit that it can come up with are then used to drive the forecast [9]. Prophet is easy to use, and it automatically outputs predictions as well as confidence intervals. In our case, to run Prophet for a single city, we simply take the historical data for a given city and rename the columns as necessary. Prophet requires the date column to be called ‘ds’ and the target variable column to be called ‘y’. We feed this data to a Prophet object to fit a model, and then create a dataframe with future predictions spanning 3 years into the future. As in the case of SARIMAX, this is the process of running the algorithm for a single city. In the next section, we will describe the steps we took to parallelize this process to run for all the cities in the data.

4.5. PySpark

Now that we have discussed two of the univariate models that we employed and how they predict future values based on historical data, we can now discuss the methods in which these models were employed on all 320 cities of the dataset. Training these models on a single machine would clearly be time-consuming. To improve performance, we chose to employ PySpark to parallelize the training process across multiple machines.

To begin, for both models discussed above, we read the data into Pandas Dataframes. This allowed us to do any preliminary preprocessing of the data such as renaming columns per the requirements of Prophet. We employed a SparkSession with Yarn as the resource manager. We then create a PySpark dataframe based on a sql query that partitions the data by city. This creates partitions of the data that can then be distributed across a compute cluster for parallel model training similar to Figure 1. We define the schema that we expect for the output.

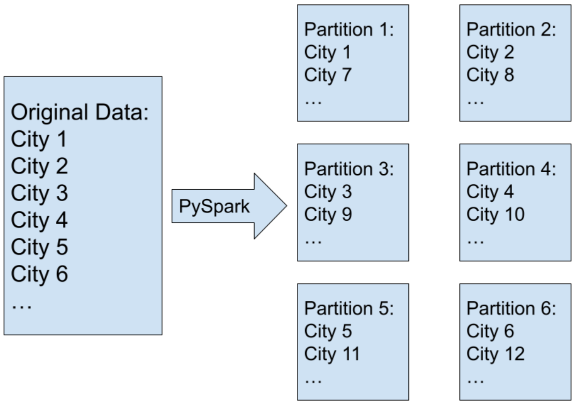


Figure 1. Demonstration of how PySpark partitions data by city.

At this point we define a Pandas UDF, or user-defined function, that defines how each worker node is to train a given model on the data partition that it has been provided. A user-defined function is a function that allows a user to tell Spark what functions to perform. Pandas UDF is an API that provides for smooth interaction between Pandas and PySpark [10]. Essentially, it allows you to write a Python function that you want each worker node to employ, combining the functionality of Pandas with the benefits of parallelization. It is within this UDF that the model is trained for each city, with the result returned as a PySpark dataframe. The resulting dataframe could then be saved as any format, such as a CSV file. We chose to save it to GCP BigQuery to allow for flexible SQL query processing against the data. We will discuss our use of BigQuery in a later subsection.

4.6. Google Cloud Platform

Now that we have discussed how we were able to code our models to take advantage of parallelization, we can talk about the compute cluster that we set up to run our univariate models. Due to prior familiarity with Google Cloud Platform, we opted to go with that route. Given the fact that we were using GCP under a free trial, we were limited to 8 CPU cores. Therefore, to run our models, we deployed a Dataproc cluster with 1 master node and 3 worker nodes, each with 2 CPU cores. We selected the “Enable Component Gateway” option to allow for easy viewing of the PySpark job status. We deployed Series N2 machines of type n2-standard-2 (2 vCPU, 8 GB memory) for both the master and worker nodes.

An important step that we learned about in this process is that you need to preload your nodes with any pip packages necessary to execute your code. We did this by adding “Cluster properties”. Specifically, we deployed our cluster with the following pip packages installed:

Cluster Properties:

- Prefix: Dataproc
- Key: pip.packages
- Value:
  - numpy==1.23.5
  - prophet==1.1.2
  - pmdarima==2.0.3

The cluster created using these settings had all the packages needed to run both the Prophet and SARIMAX models on our data as shown in Figure 2. At this point, it was just a matter of submitting the appropriate job to PySpark based on the model we wanted to use to make our predictions.

At the end of each job, we chose store the output data in Google Cloud Platform’s BigQuery service. This choice gave us the flexibility to explore the resulting data using the power of SQL. We could easily filter the results and perform calculations such as Mean Standard Error. An overview of the process can be seen in Figure 3. We will discuss the results of our predictions in a later section. This concludes the methodology and tools we used to run the univariate models on all the cities in our data.

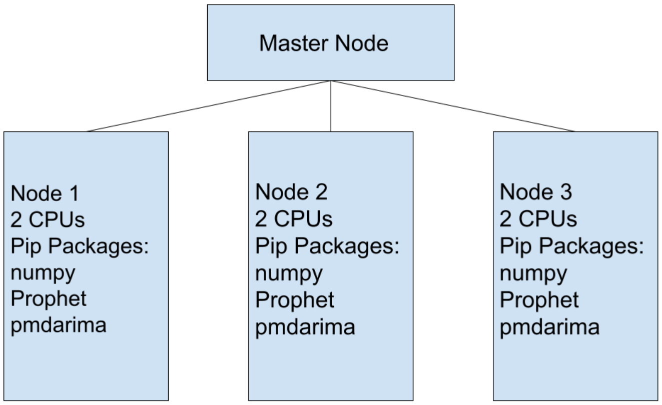
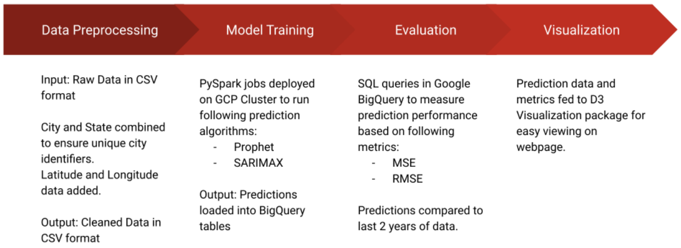


Figure 2. GCP Cluster Setup for Pyspark Jobs



Input: Raw Data in CSV format

City and State combined to ensure unique city identifiers.
Latitude and Longitude data added.

Output: Cleaned Data in CSV format

PySpark jobs deployed on GCP Cluster to run following prediction algorithms:
- Prophet
- SARIMAX

Output: Predictions loaded into BigQuery tables

SQL queries in Google BigQuery to measure prediction performance based on following metrics:
- MSE
- RMSE

Predictions compared to last 2 years of data.

Prediction data and metrics fed to D3 Visualization package for easy viewing on webpage.

Figure 3. Diagram of Data Lifecycle

4.7. LSTM

LSTM (Long Short-Term Memory) is a recurrent neural network (RNN) architecture that is used for processing sequential data. It is effective in handling time series data, where the previous inputs are dependent on the current inputs. LSTM networks use a memory cell to store information about previous inputs and selectively forget or remember information based on the input and output. The main advantage of using LSTM for time series prediction is its ability to capture long-term dependencies between input data points. However, We have encountered some of the LSTM cons. To use LSTM for daily average temperature prediction one of the main issues is that LSTM models can be quite complex and require a large amount of training data to learn effectively. Also, the LSTM Model can be slow to train and require a significant amount of computational resources. We note that compared to other data series prediction models LSTM was the slowest. Another issue is that LSTM models can suffer from overfitting, where they memorize the training data too well and fail to generalize to new data. Our data for each city is different and some cities were causing the model to overfit and the results were not representative of the actual trends of the data. This was the biggest problem for our prediction, where the model must be able to generalize to future time steps. In conclusion, for average temperature prediction using the other models was more efficient. Which are faster to train and require less data.

4.8. Linear Regression

Linear regression [15] is a statistical technique commonly used to model the relationship between a dependent variable and one or more independent variables. In our case of predicting daily average temperature, linear regression can be an effective approach because typically there is a strong linear relationship between temperature and time of day. Using linear regression, we can estimate the relationship between the date and the temperature. Linear regression is useful for predicting daily average temperature because it can handle continuous variables such as time. Also, linear regression models are easy to interpret, and can provide insight into the strength and direction of the relationship between the predictors and the outcome variable.

5. Results

This section will explain and describe our results.

5.1. Results Interpretation

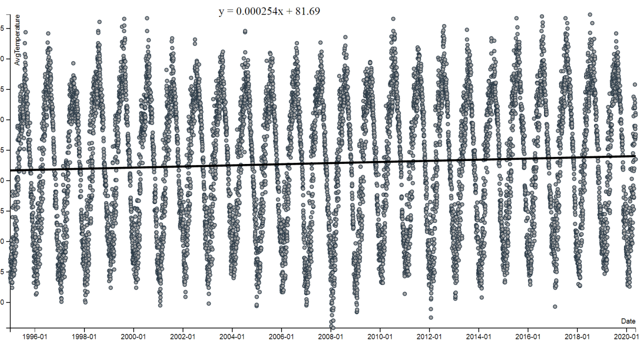


Figure 4. Linear Regression for Abu Dhabi

The linear regression line we created is in the form of

$$y = mx + b$$

where y is the dependent variable (average temperature), x is the independent variable (daily time), m is the slope of the line, and b is the y-intercept.

For example, for Abu Dhabi the slope of your regression line is 0.000254 (see Figure 4), which means that for every one unit increase in the independent variable (daily time), the dependent variable (average temperature) increases by 0.000254 units. Since we have the data for 24 years we can conclude using linear regression that there is a positive linear relationship between the average temperature and daily time. Meaning that the average temperature increases with time. The slope is small since the time is taken daily. Thus, for a specific year, the slope can be calculated as the slope for a day multiplied by the number of days in the year. 0.000254 x 365 = 0.09271. So for every year the change in average temperature for Abu Dhabi is 0.09271 Fahrenheit.

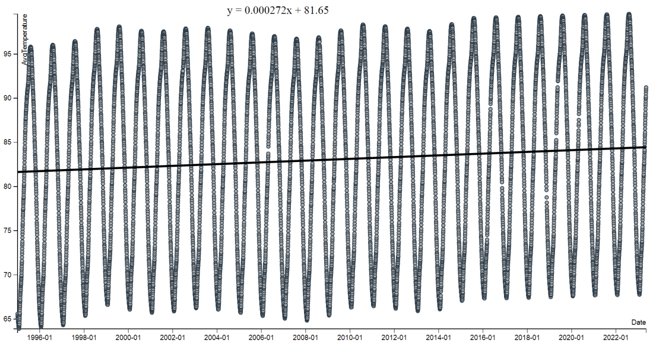


Figure 5. Linear regess of predictions

The linear regression line created by the predicted values (see Figure 5) was very similar to the original data regression line, but we now have more data since we have created predictions for two more years.

$$Orignaldata : y = 0.000254x + 81.69$$

$$Prediction : y = 0.000272x + 81.65$$

The new regression line indicates that the speed at which the temperatures are changing is increasing in the positive direction.

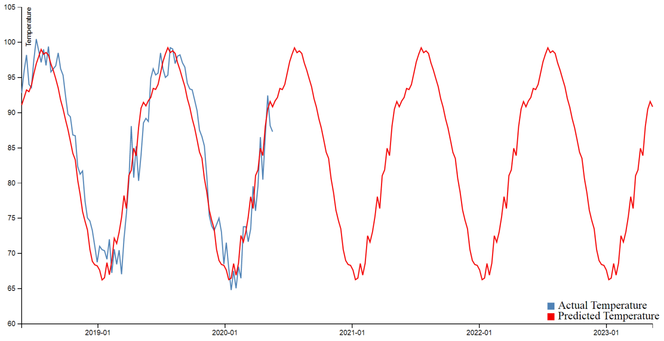


Figure 6. Data prediction using SARIMAX for Abu Dhabi

Our results for daily prediction using SARIMAX have less error rate then prophet. Since we have created our predictions for SARIMAX by aggregating the daily temps to weekly we ended up with better predictions since the change in average temperatures weekly to less than daily. As we can see in Figure 6 our Predictions are very close to the actual data.

Our results for daily prediction using Prophet are very promising (see Figure 7). Prophet created results as follow: y hat: This is the predicted value for the time period. It shows the expected value of our dependent variable (average temperature) at a specific point in time, given the historical data and the model's estimation.y lower: This is the lower bound of the prediction confidence interval. It shows the minimum value that the dependent variable is likely to be.y upper: This is the upper bound of the prediction interval. It shows the maximum value that the dependent variable is likely to be. The prediction interval captures the uncertainty around the predicted value, y upper represents the upper end of that interval, and y lower represents the lower end of that interval. Thus, the outputs provide a range of possible values for the predicted variable, along with an estimate of the uncertainty around the predicted value. When data is not sufficient we notice that the uncertainty or confidence interval is larger than cities that have sufficient data. Also the interval becomes larger so we are less confident when we predicate data that is too far in the future.

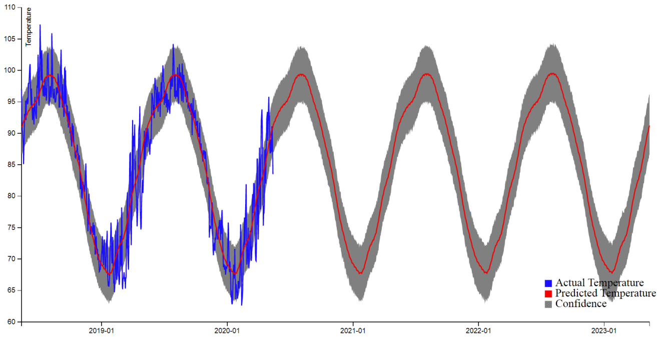


Figure 7. Data prediction using Prophet for Abu Dhabi

5.2. Research Question to Answer

For this project we wanted to see if we could use time-series forecasting to predict warming in the temperature of world cities. We used linear regression as well as the univariate time-series forecasting models SARIMAX and Prophet for predicting temperature changes in the world. The linear regression analysis shows a positive relationship between temperature and time, with a small slope indicating a small yearly temperature change. The SARIMAX model, with weekly aggregated data, has a lower error rate than the Prophet model, which provides a range of possible values for the predicted variable and an estimate of the uncertainty around the predicted value. When there is insufficient data, the confidence interval becomes larger, and the prediction becomes less reliable. Each model we used had a purpose and they all tell us that there is a positive relationship between temperature and time.Thus, all models were consistent in telling us that temperatures of earth are increasing with time.

6. Discussion

This section will discuss and interpret our results.

6.1. Results Evaluation

RMSE stands for Root Mean Square Error [14], and it is used as a metric for evaluating the performance of a

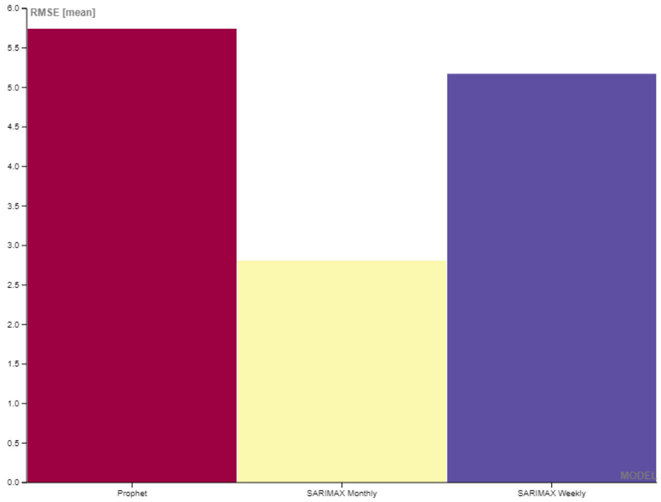


Figure 8. RMSE Results by Model

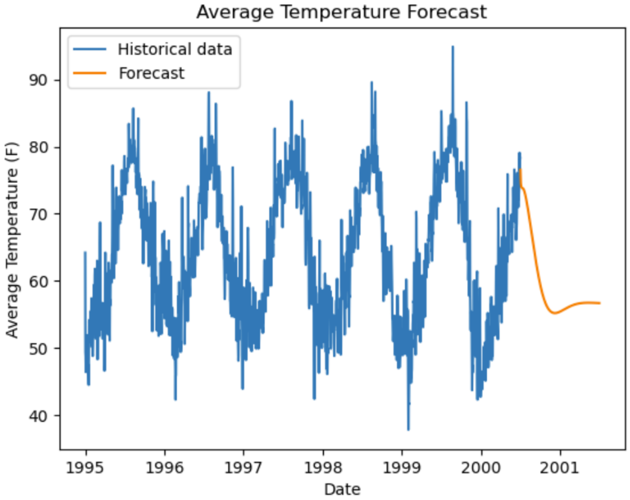


Figure 9. Data prediction using Prophet for Abu Dhabi

model. It measures the difference between the actual and predicted values of a target variable. Using the daily average temperatures, RMSE is used to measure how well a model is able to predict these temperatures based on some input features or variables. To calculate RMSE, we first calculate the squared differences between the actual and predicted values of average temperatures for each day, week, or month depending on which model we use. Then, you would take the average of these squared differences, and finally take the square root of the result. The formula [14] we used for RMSE is:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}}$$

We used Google's BigQuery to run the mathematical formula on the output data of each of the models we used. When we evaluated the models using the RMSE metric we saw that the best one with the best RMSE was the SARIMAX with monthly predictions (see Figure 8). The reason is that by using monthly our data becomes smaller so it's easier to learn the patterns and make better predictions. The SARIMAX using the weekly predictions was not as good as the monthly predictions but the results were very good. Lastly, The RMSE of the prophet model was the worst but that's because we predict daily temperatures.

When testing ARIMA on our data we encountered many limitations. First, there is quite a bit of subjectivity involved in determining (p,d,q) order of the model. We solved this limitation by using auto arima to find the best (p,d,q) for our data. Second, ARIMA has poorer performance for long term forecasts. Our data is very long. We have data of more than 24 years, but to solve the limitation we used smaller amounts of data for the ARIMA. Third, ARIMA cannot be used for seasonal time series. This is the limitation that stopped us from using normal ARIMA. The initial results showed us that ARIMA can not be used for our data predictions since our data is not stationary and it is seasonal in nature (see Figure 10).

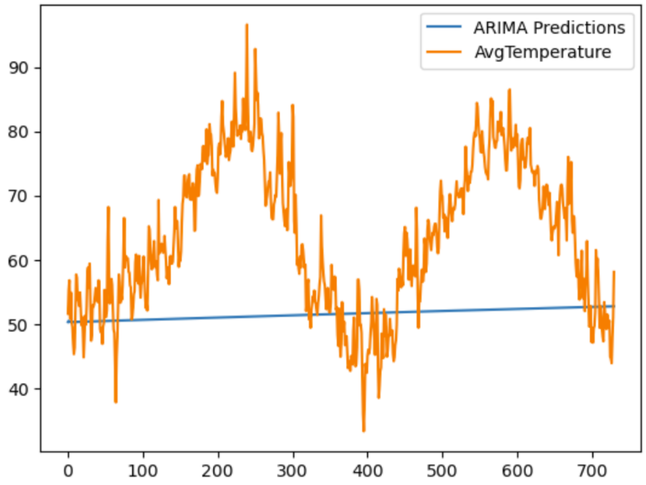


Figure 10. Data prediction using Prophet for Abu Dhabi

LSTM has a look back window, the bigger this window is the better the results are. However making this look back window big make this model even more computationally expensive to run (see Figure 9).

Our results show are consistent with existing research and theories [13]. Our results show that the temperature are increasing. Our machine learning models gave us evidence that the earth temperature will increase by around 1 F degree every 10 years or 15 years for 1 C degree.

6.2. Results Explanation

When running our models, Prophet took about 25-30 minutes to run while the two SARIMAX models, weekly and monthly, took about 3.5-4 hours. Keep in mind that

Prophet was able to manage daily data, while SARIMAX required the data to be aggregated in weekly or monthly intervals to run in a reasonable amount of time. This suggests that Prophet is the clear winner in terms of pure speed of predictions.

To measure performance, we opted to use RMSE as a metric because the results it provides are in units similar to the units of the data (degrees Fahrenheit). For monthly predictions, SARIMAX obtained an average RMSE value of 2.81. When utilizing SARIMAX for weekly predictions, SARIMAX received an average RMSE of 5.17. As for Facebook Prophet, the average RMSE was 5.74. This might lead one to suggest that SARIMAX outperformed Prophet.

But this conclusion is not exactly fair as it's not a clear apples-to-apples comparison. The monthly SARIMAX job was measured by how well it predicted the average temperature for a month, which is a much easier task than predicting the temperature for a particular day. We see that predicting the average temperature for a given week using SARIMAX resulted in an RMSE closer to that of Prophet. This makes sense in that it is harder to make accurate predictions the more granular your time window.

Therefore, taking into account the speed with which Prophet was able to make daily predictions, it came out as the preferred time-series forecasting method.

6.3. Potential Applications

Predicting temperatures can have many useful applications in various fields such as energy, agriculture, transportation, government programs, and health care. We will demonstrate each application in 7.3

7. Conclusion

This section will be the conclusion of our research project in which we summarize our main findings, explain how they contribute to the domain of Big Data, and talk of potential future improvements.

7.1. Findings and Summarization

In conclusion, the accuracy of the predictions made by our models will depend on the quality and quantity of the data used to build the model. Factors such as missing data, outliers, and non-linear relationships between variables can reduce the accuracy of the model's predictions. Therefore, each model needs the correct type of data for it to produce the best predictions. It is important to know that Not all cities are impacted at the same rates, some cities temperature will increase faster than other cities but we concluded that there is a positive relationship between temperature and time.

By leveraging machine learning models, we have gained valuable insights into the Earth's temperature patterns. We can conclude that the planet's temperature will continue to rise steadily over time, with an expected increase of approximately 1°F per decade.

7.2. Research Question & Problem Statement

The research question attempted to be answered in this project was what the future temperature of a city would be in 2 years. This research question was addressed by developing multiple Machine Learning Models, specifically Facebook Prophet, ARIMA (Autoregressive Integrated Moving Average), SARIMAX (Seasonal ARIMA), and LSTM (Long Short-Term Memory). These models were trained with past data in order to predict the future temperature of a city in 2 years. After all the models were trained, evaluation metrics, such as RMSE (Root Mean Squared Error) were calculated to determine how accurate each model was as well as to determine which model should be chosen to trust for the predictions.

7.3. Project Importance and Potential Impact

Predicting temperatures have significant importance and impact in various fields such as energy, agriculture, transportation, government programs, and health care.

- Energy: predicting the average temperature can help energy companies manage their power grids and predict energy demand since people use more power when temperature is higher. This can help avoid power outages and reduce energy costs.
- Agriculture: predicting the average temperature can help adjust watering schedule for framers. It will also help framers find the best type of crop to plant since some crops can't stand high temperatures. Thus, Farmers can use the average temperature predictions to determine the best planting and harvesting times, reduce water usage, and increase crop yields.
- Transportation: predicting the average temperature can help transportation companies in planning their schedule of operations. For example, airlines can adjust their flights based on weather predictions to avoid delays and cancellations caused by extreme weather conditions.
- Government Programs:predicting the average temperature can be used in government programs such as disaster management and emergency planning, and incentive programs for using clean energy. temperature predictions can help predict natural disasters such as heatwaves, hurricanes, and tornadoes.That will allow governments to prepare for a respond accordingly.
- Health Care:predicting the average temperature can be useful in the health care industry to help prevent and treat temperature-related illnesses. For example, hospitals can prepare for an influx of patients during heatwaves. Also, temperature predictions can help researchers identify and monitor disease outbreaks related to temperature changes.

7.4. Future Improvements

To improve our models we need to use bigger systems with more resources to train the data. We used CPU's for our training but our models were made to work better with GPU's. Also getting data for more recent years should improve the performance of the models. Using more changing features such as humidity, and wind speed they might improve the accuracy of the models.

Acknowledgments

The authors would like to thank Dr.Pegah Ojaghi and this research project assignment as not only did we learn about the theory of Big Data, but we also learned how to work with Big Data and the multiple applications Big Data has.

References

[1] M. Auhl, Towards Data Science, *What is an ARIMA Model?*, https://towardsdatascience.com/what-is-an-arima-model-9e200f06f9eb

[2] A. Hayes, Investopedia, *Autoregressive Integrated Moving Average (ARIMA) Prediction Model*, https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp

[3] S. Wu, Towards Data Science, *Stationarity Assumption in Time Series Data*, https://towardsdatascience.com/stationarity-assumption-in-time-series-data-67ec93d0f2f

[4] A. Bajaj, Neptune.ai, *ARIMA & SARIMA: Real-World Time Series Forecasting*, https://neptune.ai/blog/arima-sarima-real-world-time-series-forecasting-guide#:~:text=ARIMA%20and%20SARIMA %20are%20both,into%20account%20any%20seasonality%20patterns.

[5] T. Smith, Alkaline.ml, *API Reference: pmdarima*, https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html

[6] StatsModels, Github, *SARIMAX model too large #5727*, https://github.com/statsmodels/statsmodels/issues/5727

[7] M. Krieger, Towards Data Science, *Time Series Analysis with Facebook Prophet: How it works and How to use it*, https://towardsdatascience.com/time-series-analysis-with-facebook-prophet-how-it-works-and-how-to-use-it-f15ecf2c0e3a

[8] Facebook Prophet, Facebook Github.io, *Prophet: Forecasting at Scale*, https://facebook.github.io/prophet/

[9] A. Choudhary, Analytics Vidhya, *Generate Quick and Accurate Time Series Forecasts using Facebook's Prophet (with Python & R codes)*, https://www.analyticsvidhya.com/blog/2018/05/generate-accurate-forecasts-facebook-prophet-python-r/

[10] R. Jackson, Medium, *Modeling at Scale with Pandas UDFs (w/ Code Example)*, https://medium.com/@t7jackso/modeling-at-scale-with-pandas-udfs-code-example-6cb8ef572476

[11] statsmodels.org, statsmodels, *Seasonal AutoRegressive Integrated Moving Average with eXogenous regressors model#5727*, https://www.statsmodels.org/devel/generated/statsmodels.tsa .statespace.sarimax.SARIMAX.html

[12] statsmodels.org, statsmodels, *Autoregressive Integrated Moving Average (ARIMA) model, and extensions #5727*, https://www.statsmodels.org/devel/generated/statsmodels.tsa.arima. model.ARIMA.html#statsmodels.tsa.arima.model.ARIMA

[13] J. Garthwaite, Stanford News, *Earth likely to cross critical climate thresholds even if emissions decline, Stanford study finds*, https://news.stanford.edu/2023/01/30/ai-predicts-global-warming-will-exceed-1-5-degrees-2030s/#: :text=The%20study%2C%20published%20Jan. ,fall%20in%20the%20coming%20decade

[14] Singla, P., Duhan, M., & Saroha, S. (2022). Different normalization techniques as data preprocessing for one step ahead forecasting of solar global horizontal irradiance. In Artificial Intelligence and Solar Energy Systems: Advances in Science and Engineering (pp. 209-230). Elsevier. doi:10.1016/B978-0-323-90396-7.00004-3

[15] Schneider, A., Hommel, G., & Blettner, M. (2010). Linear regression analysis: part 14 of a series on evaluation of scientific publications. *Deutsches Ärzteblatt International*, 107(44), 776-782. doi: 10.3238/arztebl.2010.0776. PMID: 21116397; PMCID: PMC2992018.

8. Explanation of Background and Learnings

I have taken classes in machine learning, Python, and data structures. However, all of the topics that we have discussed and used in this project, such as PySpark, GCP, and D3, I learned from this class. I have never taken any class that teaches these tools before. The courses offered at SDSU only teach the basics. For instance, the machine learning class I took focused more on how AI works mathematically rather than applying it to real-life situations. I learned how to use all of these tools from this class. I hope that the project reflects the amount of effort I have put into learning these topics. I'm also very interested in learning about big data, so I spent a lot of time studying. I was able to complete all of the assignments on time because I put in a lot of effort to learn all of the topics. The project and the homework have been challenging, but it has also been rewarding. I was able to overcome these challenges by always asking questions in class when we encounter a new topic, and to practise it when I have time. Also, I have invested a significant amount of time and effort into studying every small detail, and I hope that this is reflected in the my work whether it's the homework or this project.