## Exercise 1 Report

**Akash Shingha Bappy [2307938]**

**1. My Learning**: Today, from this exercise I learned about Spark which is a cluster based computer system that includes a library to process big data. I also learned some basic python commands to process big data such as creating a session, defining a data frame, opening data both locally and with url, saving/writing to a file, accessing data with sql like queries such as selecting, sorting, rounding different data. Finally I learned to visualize data as images with matplotlib and pandas library which I used later to visualize two different figures.

**2. Difficulties and Problems:**  Although the instructions were quite straightforward, I faced some issues while proceeding with the instruction. These are described below.

* I was unable to open the data from the url. Opening data from local storage was done successfully with *“df = spark.read.csv("file\_path")”*. But it failed to parse the data when the path was replaced with a url. I tried to solve it by taking help from the following [stackoverflow](https://stackoverflow.com/questions/69330177/pyspark3-read-file-from-https-url) and [databricks](https://community.databricks.com/t5/data-engineering/read-csv-directly-from-url-with-pyspark/td-p/12053) links. But that didn’t work correctly as the data was not well arranged and I could not resolve that.
* I have made a figure showing air\_temperature vs wind\_speed on a monthly basis. However, for this I needed to convert the dateTime from string to date format with the help of pandas.In the instruction there was a mention of “*pd.to\_numeric()”* to convert string to number, but it gave a blank figure. It was difficult for me as it was not mentioned in the instruction. However, with the help of [stackoverflow](https://stackoverflow.com/questions/25146121/extracting-just-month-and-year-separately-from-pandas-datetime-column) and [pandas documentation](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_datetime.html) I was able to do it with “*dff['dateTime'] = pd.to\_datetime(dff['dateTime'])”*

**3. Figures**: figure 1 shows the wind speed with respect to air temperature. It can be seen that wind speed was higher around 10℃ and gradually decreased at higher or lower temperatures. However there are some outliers for 0℃. On the other hand, In figure 2 average wind speed is displayed corresponding to the months of different years. It is visible that it was the lowest in July which is summer and highest in December that is winter.

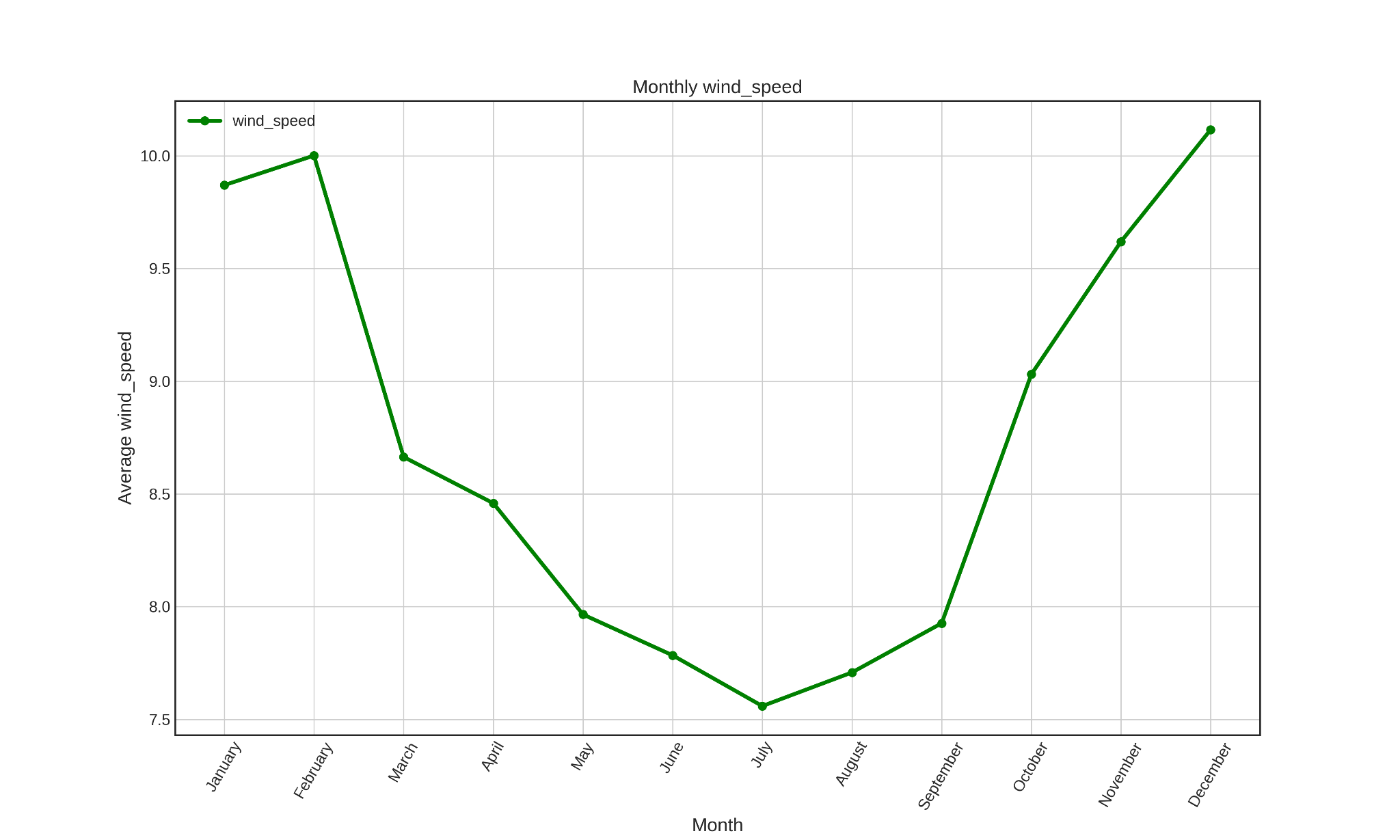
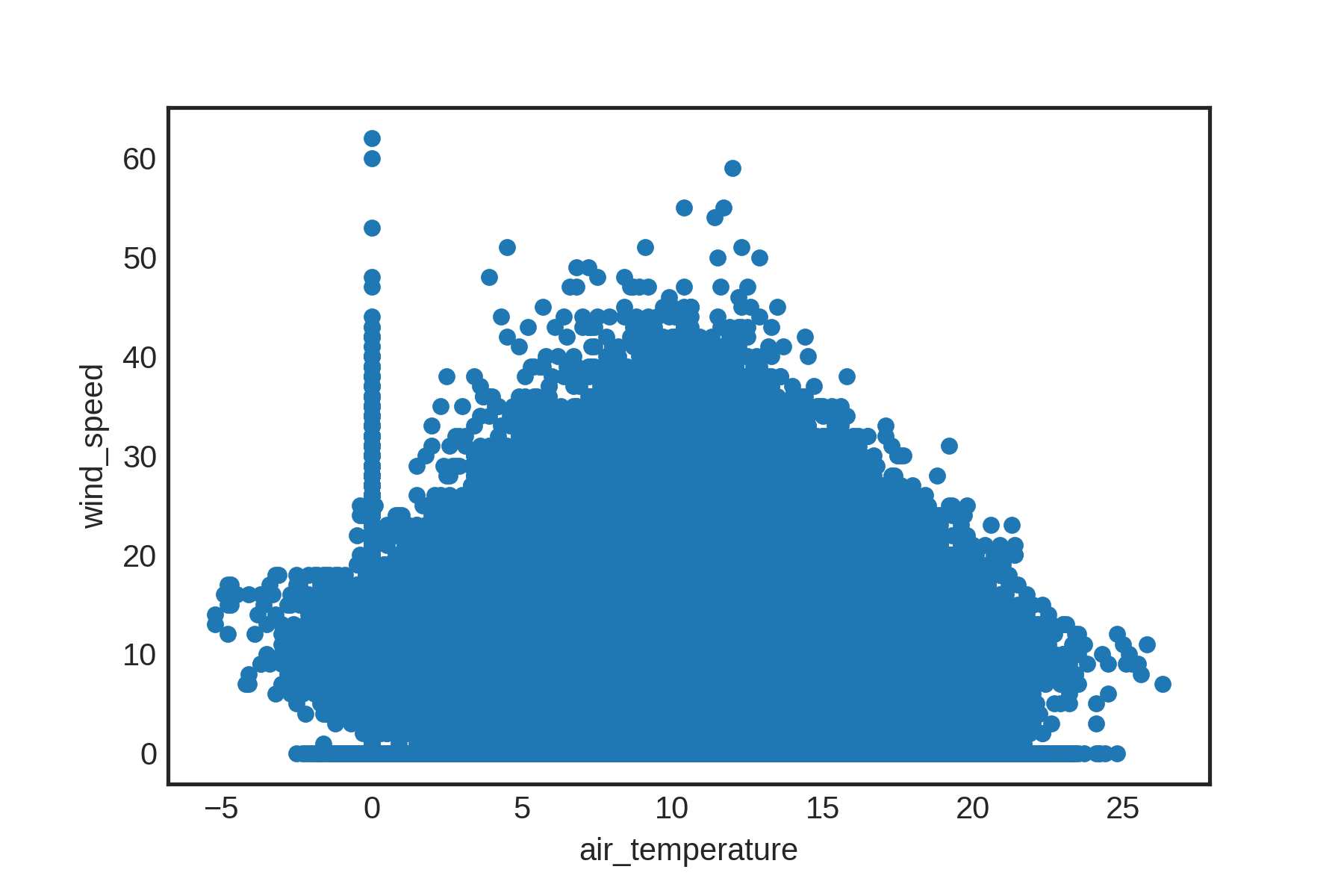


Figure 1: wind speed vs air temperature Figure 2: average wind speed vs Month

**Reference**:

1. <https://stackoverflow.com/questions/69330177/pyspark3-read-file-from-https-url>
2. <https://community.databricks.com/t5/data-engineering/read-csv-directly-from-url-with-pyspark/td-p/12053>
3. <https://stackoverflow.com/questions/25146121/extracting-just-month-and-year-separately-from-pandas-datetime-column>
4. <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_datetime.html>

## Exercise 2 Report

**Akash Shingha Bappy [2307938]**

1. **My Learnings:**

In this exercise, I learned about Structured Streaming which is a high-level streaming API by Spark. It was introduced in Spark 2.2, which offers higher-level optimizations, event-time processing, and support for continuous processing. It allows data streaming by continuously updating tables, making stream processing very similar to batch processing. It includes several terminologies like Input Source, Sink, and Output Mode, each with several built-in options which was described in the document nicely. Structured Streaming APIs also include readStream and writeStream operations for reading and writing streaming data. Additionally, basic operations like filtering can be applied to streaming data similarly to batch data.

Like the previous [exercise](https://moodle.oulu.fi/pluginfile.php/2220353/mod_resource/content/1/Exercise%201.pdf), I built a Spark Session in Pyspark Notebook and then read the data as well as schema as instruction. Then I read the stream and passed the “*maxFilesPerTrigger*” value 1 as I wanted to read 1 file. Then I wrote the stream to a sink and queried some data.

1. **Difference between methods and operations provided for structured streaming using SparkSession and Spark StreamingContext:**

- `*SparkSession*` is typically used for batch processing and structured streaming applications. It provides a unified entry point for reading data, executing SQL queries, and performing operations on DataFrames and Datasets.

- `*SparkStreamingContext*`, on the other hand, is specifically designed for handling real-time data streams. It provides functionality for creating DStreams (Discretized Streams), which represent a continuous stream of data.

- While both `SparkSession` and `SparkStreamingContext` can be used to work with structured streaming, their primary focus and APIs differ. `SparkSession` is more oriented towards batch and structured streaming, offering DataFrame-based APIs, while `SparkStreamingContext` is focused on handling real-time data streams using DStreams.

Reference:

1. <https://moodle.oulu.fi/pluginfile.php/2220353/mod_resource/content/1/Exercise%201.pdf>
2. <https://moodle.oulu.fi/pluginfile.php/2220355/mod_resource/content/2/Exercise%202.pdf>
3. <https://dzone.com/articles/spark-streaming-vs-structured-streaming>
4. <https://youtu.be/2VRRjfNlwMQ?si=PDuTUjYxSdxEGFCe>

## Exercise 3 Report

**Akash Shingha Bappy [2307938]**

In this exercise, we were introduced to Spark MLlib which is a scalable machine-learning library that offers a wide range of algorithms optimized for distributed computing. It runs on Hadoop, Apache Mesos, Kubernetes, standalone, or in the cloud, against diverse data sources. MLlib supports model building, hyperparameter tuning, and evaluation, making it ideal for big data machine learning tasks.

In machine learning, ML Pipelines are like pathways that help us build and train models efficiently. One really good thing about using ML Pipelines is that they help us tune the settings, called hyperparameters, of the models. For example, In our ExampleData dataset, there are lots of different features like date, humidity, wind speed, and others where the task was to predict something, like air temperature, based on these features.

Now, tuning hyperparameters is like finding the perfect settings for the model. It's super important because it can make the model perform much better. For example, if we're predicting air temperature, we want the model to be as accurate as possible so that we can make a better prediction. Hyperparameters control how the model learns from the data, and finding the best ones can make the predictions much more accurate.

In Spark, there are tools called ML Pipelines that help us with this. As the accuracy of a model varies on different data based on the parameters manually tweaking it takes a lot of time and effort. With ML Pipeline, they let us try out different combinations of hyperparameters automatically, saving us a ton of time and effort. So, it makes it easier for us to find the best settings for the models, which leads to more accurate predictions.

In our specific case, where we are trying to predict air temperature from features like date, humidity, wind speed, and msl, using ML Pipelines with models like Gradient-Boosted Trees or Random Forests is really helpful. These models are powerful, but their performance depends a lot on having the right hyperparameters set. With the right environment and data, it can assure good efficiency, consistency, scalability and Reproducibility.

After training the models with different hyperparameters using ML Pipelines, we can evaluate them using metrics like RMSE (Root Mean Squared Error), R-square, and MAE (Mean Absolute Error) to see which one performs the best. Then, we can use libraries like Seaborn and Matplotlib to visualize the results, making it easier to understand how well the models are doing and which one is the best for predicting air temperature.

Therefore, it can be said that hyperparameter optimization is indeed a big benefit of using ML Pipeline as with proper utilization, it can ensure great accuracy in less time and effort.

Reference

1. [MLlib | Apache Spark](https://spark.apache.org/mllib/)
2. [ML Pipelines - Spark 3.5.1 Documentation (apache.org)](https://spark.apache.org/docs/latest/ml-pipeline.html)
3. [Classification and regression - Spark 3.5.1 Documentation (apache.org)](https://spark.apache.org/docs/latest/ml-classification-regression.html)
4. [Machine Learning Pipelines: Benefits, Challenges, Use Cases (plat.ai)](https://plat.ai/blog/machine-learning-pipeline/)

## Exercise 4 Report

**Akash Shingha Bappy [2307938]**

### My Learnings:

In this exercise, I have learned about deep learning with a Multilayer Perceptron Classifier using Spark MLlib. First, a spark session was created along with importing all the necessary libraries. Then the data was loaded and preprocessed such as labeling and numerical conversion. Additionally, the data was split to training, validation and test data with a data size of 70%, 20% and 10% respectively. Then, A pipeline was constructed along with a classifier to fit with the training data. Lastly, the data was evaluated by computing matrics like weighted precision, recall, and accuracy for each set along with the confusion matrix which was plotted at the end. Overall this exercise covers the implementation of a deep learning classifier using PySpark Mlib that includes data preprocessing, model configuration, training, result evaluation and visualization.

### Result:

From steps 6 and 7, the evaluation matrices(figure 1) and confusion matrix(figure 2) were obtained which gives an insight about the performance of the model. Here, the precision and recall were quite well as a result an accuracy of over 97% was achieved for all training, validation and test sets. Along with that, from the confusion matrix, it can be said that the model is much robust as the number of misclassifications was very low (FP=104, FN=50) compared to the true classifications (TN=3760, TP=3532).

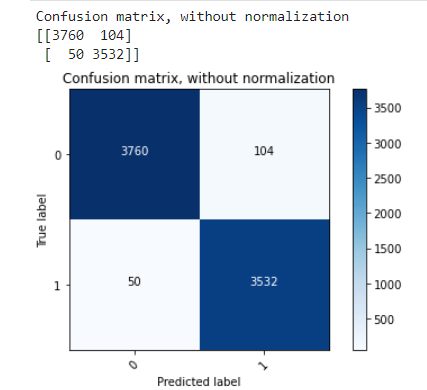
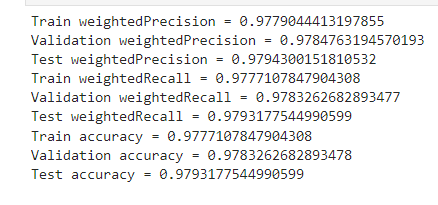
****

Figure: 1 Evaluation Matrices (Precision, Recall, Accuracy) For Train, Validation And Test Set.

Figure 2: Confusion Matrix Comparing Predicted And True Labels

**Reference:**

1. [BIg Data Processing and Application: Exercise 4.pdf](https://moodle.oulu.fi/pluginfile.php/2279005/mod_folder/content/0/Exercise%204.pdf?forcedownload=1)
2. [Evaluation Metrics - RDD-based API - Spark 3.5.1 Documentation (apache.org)](https://spark.apache.org/docs/latest/mllib-evaluation-metrics.html)