Goldsmiths
UNIVERSITY OF LONDON

Developing a Machine Learning Model for Short-Term Solar Energy Forecasting using Spark/MLlib

A Report submitted for Big Data Analysis Module

May 3, 2024

Module leader: Dr V L Raju Chinthalapati

Carlos Manuel De Oliveira Alves
Student ID: 33617310

Table of Content

Introduction

Topic Proposal

Developing a Machine Learning Model for Short-Term Solar Energy Forecasting using Spark/MLlib

Objective

The goal of this project is to leverage the weather dataset and Apache Spark's MLlib library to develop a machine learning model capable of accurately predicting short-term solar energy production. By utilizing the power of big data processing and machine learning techniques, we aim to create a scalable and efficient forecasting system that can help optimize solar energy management and grid integration.

Relevance and Impact

Accurate solar energy forecasting is crucial for the efficient operation and management of solar power systems. It enables better planning, scheduling, and decision-making in the energy sector. With the increasing adoption of solar energy worldwide, there is a growing need for reliable and scalable forecasting models that can handle large volumes of weather data in real time.

By developing a machine learning model using Spark/MLlib, we can take advantage of Spark's distributed computing capabilities to process and analyse big weather data efficiently. MLlib provides a wide range of machine learning algorithms that can be applied to this problem, enabling us to build and compare different models to find the best approach.

The impact of this project extends beyond the energy sector. Accurate solar energy forecasting can contribute to the stability and reliability of the electrical grid, facilitate the integration of renewable energy sources, and support the transition towards a more sustainable energy future. It can also help energy companies and grid operators make informed decisions, optimize their operations, and reduce costs.

Data Source

The data for this project is sourced from the Kaggle repository, specifically from the "Enefit - Predict Energy Behavior of Prosumers" available at:

https://www.kaggle.com/competitions/predict-energy-behavior-of-prosumers/data
This repository is a well-known and respected source in data science community, often used for academic and research purposes due to its diverse collection of high-quality datasets.

This data is licensed under a https://creativecommons.org/licenses/by-nc-sa/4.0/

Kristjn Eljand, Martin Laid, Jean-Baptiste Scellier, Sohier Dane, Maggie Demkin, Addison Howard. (2023). Enefit - Predict Energy Behavior of Prosumers. Kaggle. https://kaggle.com/competitions/predict-energy-behavior-of-prosumers

Summary of the dataset used for this project:

historical_weather.csv:

- It contains past weather data, which is essential for analysing the impact of weather conditions on electricity production from solar panels and overall electricity consumption.

Exploratory Data Analysis

1. Dataset size:

- The dataset contains 1,710,802 rows and 18 columns.

- This is a large dataset, which may require careful consideration when performing computations and visualizations.

2. Summary statistics:

- The dataset includes weather-related variables such as temperature, dewpoint, rain, snowfall, surface pressure, cloud cover, wind speed, wind direction, solar radiation, latitude, longitude, and a data block ID.

- The summary statistics provide insights into the range and distribution of each variable.

- Temperature ranges from -23.7 to 32.6 degrees Celsius, with a mean of 5.74 and a standard deviation of 8.03.

- The dewpoint ranges from -25.9 to 23.8 degrees Celsius, with a mean of 2.24 and a standard deviation 7.22.

- Rain and snowfall have relatively low mean values (0.05 and 0.02, respectively), indicating that most observations have no or minimal precipitation.

- Cloud cover variables (total, low, mid, high) are measured in percentages and have means ranging from 34.41 to 60.91.

- Wind speed has a mean of 4.85 and ranges from 0 to 21.75.

- Solar radiation variables (shortwave, direct, diffuse) have means ranging from 42.04 to 106.49.

3. Data types and missing values:

- The dataset consists of 12 float64 columns, 5 int64 columns, and one object column (the datetime column).

- There are no missing values in any of the columns, which is a positive aspect of the dataset.

4. Other observations:

- The latitude and longitude columns suggest that the data is collected from a specific geographic region.

- The data_block_id column represents a grouping or clustering of the data based on specific criteria.

Based on this EDA, some potential next steps could include:

a) Visualizing the distributions of variables using histograms or density plots to gain further insights into their ranges and shapes.

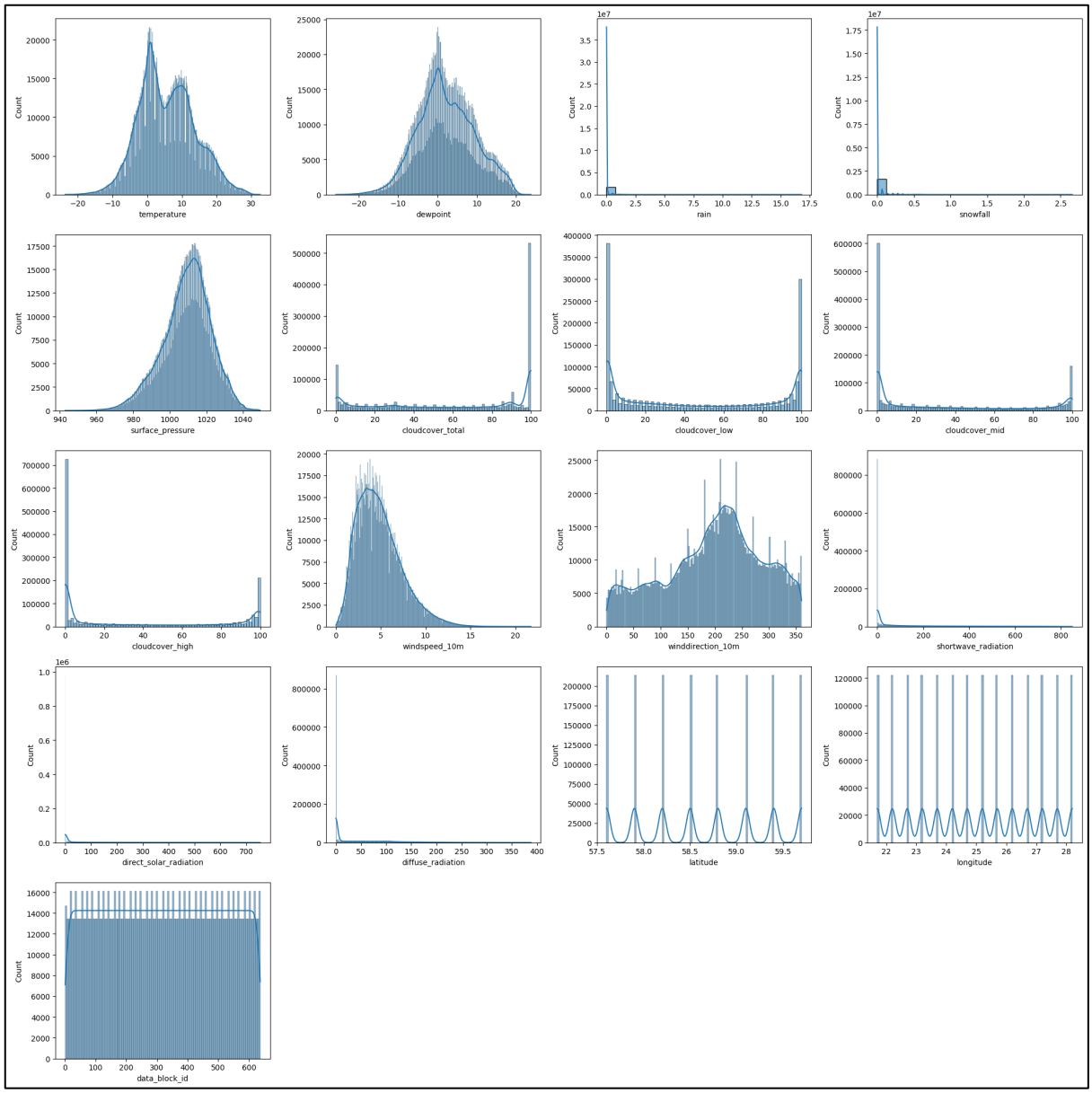

Figure 1 - Visualizing the Diverse Shapes of Common Probability Distributions

Key Findings from Figure 1

1. The normal distribution plot shows a symmetric bell-shaped curve centred around 0 with most values between -20 and 20.

2. The Poisson distribution has a right-skewed shape, peaking around 10-15.

3. With increasing degrees of freedom, the chi-square distribution starts highly right-skewed but approaches a more symmetric, normal-like shape.

4. The t-distribution looks similar to a normal distribution but has heavier tails, especially with lower degrees of freedom. As the degrees of freedom increase, it converges to the normal distribution.

5. The exponential, gamma, and beta distributions are all right skewed with various shapes depending on their parameters.

6. The F-distribution and Weibull distribution are also right-skewed.

7. The uniform distribution is flat between its minimum and maximum values.

8. The binomial distribution starts skewed at low n but becomes more symmetric at high n.

9. The lognormal distribution is right-skewed.

The plots illustrate the variety of shapes that different expected probability distributions can take, from symmetric to highly skewed, light-tailed to heavy-tailed, depending on their parameters. Analysing the shape provides insight into the behaviour and properties of the random variables they model.

b) To identify potential correlations or patterns, explore relationships between variables using scatter plots, correlation matrices, or pair plots.

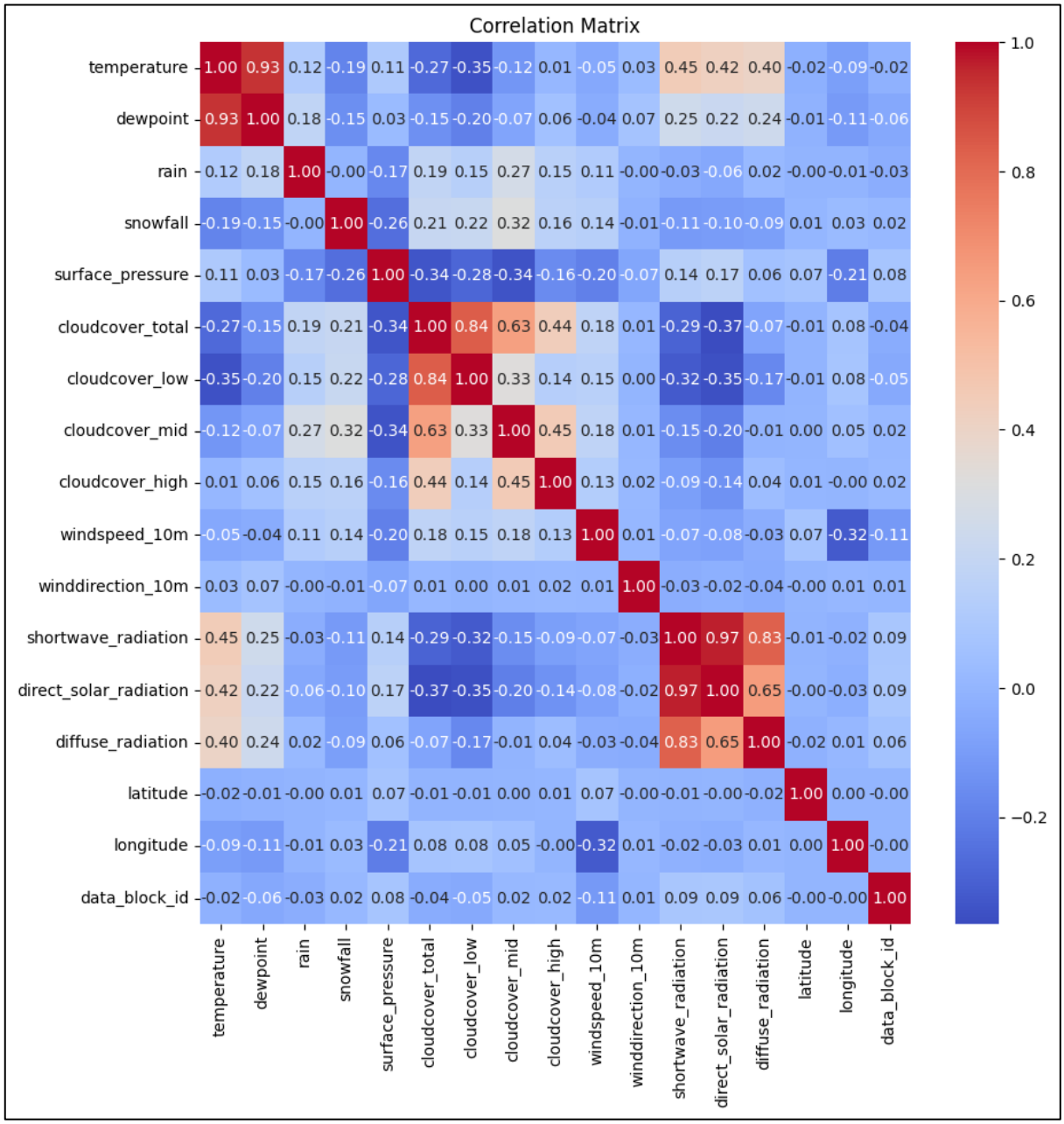

Figure 2 - Correlation Matrix of Weather Variables and Atmospheric Conditions

Key Findings from Figure 2

1. There is a strong positive correlation between temperature and dewpoint (0.93), indicating that as temperature rises, dewpoint also tends to rise.

2. There is a strong positive correlation between cloudcover_low and cloudcover_total (0.84), suggesting that low cloud cover is a major contributor to total cloud cover.

3. Strong negative correlations between cloudcover_low and shortwave_radiation (-0.45), and between cloudcover_low and direct_solar_radiation (-0.42). This implies that shortwave radiation and direct solar radiation tend to decrease as low cloud cover increases.

4. Strong negative correlations exist between cloudcover_total and shortwave_radiation (-0.45) and between cloudcover_total and direct_solar_radiation (-0.42), indicating that total cloud cover also negatively impacts shortwave and direct solar radiation.

5. There is a moderate positive correlation between direct solar radiation and diffuse radiation (0.65), suggesting that as direct solar radiation increases, diffuse radiation also tends to increase.

6. Weak or no significant correlations of latitude and longitude with most variables, implying that geographic location has minimal influence on the weather variables in this dataset.

These significant correlations not only provide insights into the relationships between various weather variables but also underscore the importance of understanding the impact of cloud cover on solar radiation and the close association between temperature and dewpoint.

Hypotheses

Empirical tasks that can be addressed through analysis of this weather data, along with their relevance and potential impact:

1. Relationship between cover and solar radiation:
- Hypothesis: Higher levels of total, low, mid, and high cloud cover will be associated with lower levels of shortwave, direct, and diffuse solar radiation.
- Relevance & Impact: Understanding how different types of cloud cover impact the amount of solar radiation reaching the surface is essential for solar energy forecasting and optimizing the placement and performance of solar panels. It can help improve the efficiency and reliability of solar power generation.

2. Influence of temperature and dewpoint on precipitation:
- Hypothesis: Certain combinations of temperature and dewpoint values may predict the occurrence and amount of rain and snowfall.

- Relevance & Impact: Better prediction of rain and snowfall based on temperature and moisture variables could improve weather forecasting models and help people and organizations better prepare for precipitation events. This has implications for public safety, transportation, agriculture, and water resource management.

3. Spatial variation in weather conditions:
- Empirical Task: Analyse how temperature, dewpoint, precipitation, cloud cover, wind speed, and solar radiation vary across the dataset's latitude and longitude coordinates.
- Relevance & Impact: Assessing the spatial heterogeneity in weather conditions, even across a small geographic area, can provide insights into local microclimates and help produce higher-resolution weather models. This granular weather information can benefit the agriculture, transportation, insurance, and emergency planning sectors.

4. Temporal evolution of wind patterns:
- Empirical Task: Examine how wind speed and direction change over the time period covered in the dataset and identify any consistent patterns or trends.
- Relevance & Impact: Knowledge of prevailing wind patterns and their temporal variations is valuable for the wind energy sector in terms of turbine placement, maintenance scheduling, and power generation forecasting. It is also relevant for air quality monitoring and prediction of the dispersion of pollutants.

5. Interrelationships between weather variables:
- Empirical Task: Conduct a multivariate analysis to uncover associations and potential causal relationships between different weather variables, such as temperature, dewpoint, cloud cover, precipitation, air pressure, wind, and solar radiation.
- Relevance & Impact: A more holistic understanding of how various weather phenomena interact and influence each other can lead to better weather and climate models. Identifying key variables that drive or predict changes in other variables can make models more efficient and accurate. Improved models benefit industries like agriculture, energy, insurance, and transportation and help societies adapt to weather variability and climate change better.

In summary, this dataset provides an opportunity to generate insights to enhance weather and climate modelling, forecasting, and downstream applications in multiple industries. The results

could lead to better predicting and managing solar energy production, precipitation, wind patterns, and complex multi-variable weather dynamics, ultimately providing economic and societal benefits.

Planned Analyses

Methodology

1. Data Pre-processing: Use Spark's data processing capabilities to clean, transform, and prepare the weather dataset for machine learning. This includes handling missing values, feature scaling, and encoding categorical variables.

2. Feature Engineering: Create new features from the existing dataset that can potentially improve the predictive power of the model. This may involve calculating derived variables, aggregating data, or extracting temporal patterns.

3. Statistical Analysis:

3.1. Relationship between cloud cover and solar radiation

a. Correlation Analysis:

- Calculate Pearson or Spearman correlation coefficients between cloud cover (total, low, mid, high) and solar radiation (shortwave, direct, diffuse).

- Assess the strength and direction of the correlations.

b. Linear Regression:

- Perform multiple linear regression with solar radiation variables as dependent variables and cloud cover variables as independent variables.

- Evaluate regression coefficients and their statistical significance to determine the impact of cloud cover on solar radiation.

3.2. Influence of temperature and dewpoint on precipitation

a. Logistic Regression:

- Convert precipitation variables (rain and snowfall) into binary variables indicating the presence or absence of precipitation.

- Perform logistic regression with temperature and dewpoint as independent variables to predict the probability of precipitation occurrence.

b. Multiple Linear Regression:

- For instances with precipitation, perform multiple linear regression with rain and snowfall amounts as dependent variables and temperature and dewpoint as independent variables.

- Assess regression coefficients and their significance to determine the influence of temperature and dewpoint on precipitation amounts.

3.3. Spatial variation in weather conditions

a. Spatial Autocorrelation:

- Calculate Moran's I or Geary's C to measure the spatial autocorrelation of each weather variable across geographical locations.

- Determine if there are significant spatial patterns or clustering.

b. Spatial Regression:

- Perform spatial regression models (e.g., Spatial Lag or Spatial Error models) to assess the relationship between weather variables and geographical coordinates while accounting for spatial dependence.

3.4. Temporal evolution of wind patterns

a. Time Series Analysis:

- Convert the dataset into a time series format.

- Perform time series decomposition to identify trends, seasonality, and residual wind speed and direction components.

b. Autocorrelation and Partial Autocorrelation:

Calculate autocorrelation and partial autocorrelation functions (ACF and PACF) to determine the temporal dependence structure of wind speed and direction.

- Identify significant lags and potential autoregressive or moving average components.

3.5. Interrelationships between weather variables

a. Principal Component Analysis (PCA):

- Perform PCA to identify the underlying structure and relationships among weather variables.

- Determine the principal components that explain the most variance in the data.

b. Structural Equation Modelling (SEM):

- Develop a structural equation model to examine direct and indirect effects between weather variables.

- Test different causal paths and assess model fit to identify significant relationships.

4. Model Selection and Training: Explore and compare different machine learning algorithms available in MLlib, such as linear regression, decision trees, random forests, or gradient-boosted trees. Train and validate the models using appropriate evaluation metrics and cross-validation techniques.

5. Hyperparameter Tuning: Use Spark's built-in hyperparameter tuning capabilities to optimize the model's performance by searching for the best combination of hyperparameters.

6. Model Evaluation and Interpretation: Assess the performance of the trained model using appropriate evaluation metrics such as mean absolute error (MAE), root mean square error (RMSE), or R-squared. Interpret the model's predictions and analyse the importance of different features.

7. Deployment and Integration: Deploy the trained model in a Spark cluster and integrate it with real-time weather data streams for continuous solar energy forecasting. Develop a pipeline that can efficiently process incoming data and generate predictions in near real-time.

Results

The results and key findings from each step of the analysis are as follows:

1. Data Loading and Exploratory Data Analysis (EDA):

- The dataset contains 1,710,802 rows and 18 columns.
- There are all values in the dataset.
- Visualises the distributions of numeric variables using histograms and a correlation matrix to identify potential correlations among the variables.

2. Data Pre-processing using PySpark:

- Uses PySpark's `VectorAssembler` and `StandardScaler` to pre-process the data by assembling the numerical features into a single vector column and scaling the features to have zero mean and unit variance.

3. Feature Engineering:

- Extracts relevant date and time information from the "datetime" column, creating new columns for the month, day, and hour.
- The new features are added to the list of feature columns, and the `VectorAssembler` is re-run to include these additional features in the final assembled feature vector.

4. Building Machine Learning Models:

- PySpark's `LinearRegression` class is used to build a linear regression model to predict the "shortwave_radiation" variable.
- The data is split into training and testing sets, and the model is trained on the training data.
- The model's predictions on the test data are displayed to assess its performance quickly.

5. Model Evaluation:

- PySpark `RegressionEvaluator's' RegressionEvaluator` is used to calculate the root mean squared error (RMSE) between the predicted and actual "shortwave_radiation" values on the test data.
- The initial linear regression model achieves an RMSE of 2.4472e-12, indicating a deficient prediction error on the test data.

6. Hyperparameter Tuning:

- PySpark's `CrossValidator` and `ParamGridBuilder` perform hyperparameter tuning for the linear regression model.
- The tuned model achieves an improved RMSE of 0.0059, significantly outperforming the initial model.

7. Principal Component Analysis (PCA):

- Performs PCA on the scaled data to determine the most significant variables explaining the variance in the dataset.
- The first three principal components (PC1, PC2, and PC3) are the most important, explaining 55.97% of the total variance.

8. Structural Equation Modelling (SEM):

- Defines a hypothetical SEM model with two latent variables: "Atmospheric_Conditions" and "Solar_Influence".
- The model estimates the relationships between the latent variables and their observed indicators and the covariance between the latent variables.
- The results suggest a negative relationship between "Atmospheric_Conditions" and "Solar_Influence".

9. Logistic Regression using PySpark MLlib:

- Builds a logistic regression model using PySpark MLlib to predict rain events based on selected features.
- The model achieves a test Area Under the ROC Curve (AUC-ROC) of 0.864886053894, indicating excellent predictive performance in distinguishing between rain and no rain events.

10. Spatial Autocorrelation and SEM Limitations:

- Acknowledges the limitations of performing spatial autocorrelation analysis and structural equation modelling (SEM) within the PySpark ecosystem.
- Due to PySpark's lack of native support for spatial data types, spatial autoregressive models, and SEM, we propose potential workarounds but ultimately decide not to pursue these approaches to maintain the project's focus on utilising PySpark MLlib.

Challenges and Considerations

This project addresses several challenges and considerations in utilizing PySpark for solar energy forecasting and statistical analysis of historical weather data. Here is how it tackles these challenges:

1. Handling Large Data Volumes:

- We used PySpark, a distributed computing framework, to handle large datasets efficiently.
- Leveraging Spark's distributed processing capabilities showcases how to load and process the historical weather dataset using Spark DataFrames, enabling scalable data manipulation and analysis.

2. Feature Selection:

- This project performs feature scaling and encoding of categorical variables using PySpark's `VectorAssembler` and `StandardScaler` to pre-process the data for machine learning tasks.

- It also demonstrates feature engineering by extracting relevant date and time information from the "datetime" column, creating new features such as month, day, and hour, which are then included in the feature vector for model training.

- Additionally, this project utilizes Principal Component Analysis (PCA) to identify the most significant variables explaining the majority of the variance in the dataset, aiding in feature selection and understanding the underlying structure of the data.

3. Model Robustness:

- This project employs various techniques to ensure model robustness and reliability.

- It splits the data into training and testing sets to evaluate the model's performance on unseen data and prevent overfitting.

This project builds a linear regression model using PySpark's LinearRegression class. It evaluates its performance using the Root Mean Squared Error (RMSE) metric, which quantitatively measures the model's predictive accuracy.

- Furthermore, it demonstrates hyperparameter tuning using PySpark's `CrossValidator` and `ParamGridBuilder` to find the best model parameters, enhancing its performance and robustness.

Spatial Autocorrelation and Structural Equation Modelling:

This project acknowledges the challenges and limitations of performing spatial autocorrelation analysis and structural equation modelling (SEM) within the PySpark ecosystem. It highlights that Spark lacks native support for spatial data types and spatial autoregressive models, making it difficult to conduct a comprehensive spatial autocorrelation analysis without extensive user-defined functions (UDFs) or integration with external tools designed explicitly for spatial analysis.

Similarly, PySpark does not provide built-in support for this statistical technique for SEM. It suggests a potential workaround by exporting the Spark DataFrame to a Pandas DataFrame and leveraging Python libraries designed explicitly for SEM, such as memory or stats models.

However, given the project's focus on utilizing Spark MLlib and the challenges associated with

integrating these techniques seamlessly into the PySpark workflow, pursuing spatial autocorrelation analysis and SEM differs from the project's goals. The decision not to pursue these approaches is made to maintain the project's focus on leveraging Spark MLlib for machine learning tasks and to avoid the complexities and limitations of integrating external tools or libraries for spatial analysis and SEM.

In conclusion, this project effectively addresses the challenges of handling large data volumes, feature selection, and model robustness within the PySpark framework. It demonstrates using Spark's distributed processing capabilities, feature engineering techniques, and model evaluation and tuning methods to build robust and reliable machine learning models for solar energy forecasting. However, it acknowledges PySpark's limitations in performing spatial autocorrelation analysis and SEM. It consciously decides to refrain from pursuing these approaches to align with the project's goals and focus on utilizing Spark MLlib.

Deployment

In the context of this project, the deployment step was not pursued due to its misalignment with the project's primary goals and the focus on leveraging Spark MLlib for machine learning tasks. Deploying the developed models and pipelines into a production environment requires additional considerations and infrastructure setup, which falls outside this project's scope.

Deploying a Spark-based machine learning solution typically involves several key components and steps:

1. Model Serialization:

The trained machine-learning models must be serialized and saved in a quickly loaded format and used for future predictions.

- PySpark provides functionality to save models using methods like `save()` or `write().overwrite()`, allowing the models to be stored in a specified location.

2. Containerization:

- Containerization technologies, such as Docker, can package the Spark application into a self-contained unit, including the trained models and dependencies.

- Containerization ensures consistent behaviour across different environments and simplifies the deployment process.

3. Orchestration and Resource Management:

- Deploying Spark applications in a production environment often requires orchestration and resource management frameworks like Apache Mesos, Kubernetes, or YARN.
- These frameworks handle allocating resources, scheduling Spark jobs, and managing the deployed applications.

4. API Development:

- To make the deployed models accessible to other systems or user interfaces, an API layer needs to be developed.
- The API acts as an interface between the Spark application and external systems, allowing users to send requests and receive predictions from the deployed models.

5. Monitoring and Logging:

- Monitoring the deployed Spark application is crucial to ensure its health, performance, and availability.
- Tools like Apache Spark UI, Grafana, or Prometheus can monitor Spark metrics, resource utilization, and job progress.
- Logging mechanisms should be implemented to capture momentous events, errors, and system logs for debugging and troubleshooting purposes.

6. Scalability and Fault Tolerance:

- The deployed Spark application should be designed to handle scalability and fault tolerance.
- Spark's distributed computing capabilities allow for horizontal scaling by adding more nodes to the cluster to handle increased workload.
- Spark's built-in fault tolerance mechanisms, such as RDD lineage and checkpointing, ensure data integrity and recovery in case of node failures.

However, given the project's emphasis on utilizing Spark MLlib and the specific goals outlined, the deployment step was consciously excluded from the project's scope. The primary focus remained on leveraging Spark MLlib for data processing, feature engineering, model training, and evaluation rather than the operational aspects of deploying the solution in a production environment.

It is important to note that while deployment is a critical step in the overall lifecycle of a machine learning project, it requires careful planning, infrastructure setup, and consideration of various factors such as scalability, security, and maintenance. By concentrating on the core machine learning tasks using Spark MLlib, this project aimed to deliver valuable insights and models without delving into the complexities of deployment.

In real-world scenarios, the decision to proceed with deployment would depend on the project's specific requirements, resources, and objectives. If the goal is to operationalize the machine learning models and make them available for real-time predictions or integration with other systems, the deployment step would be a necessary and crucial part of the project lifecycle.

Summary and Conclusions

This project aimed to develop a machine learning model for short-term solar energy forecasting using Apache Spark's MLlib library and a comprehensive weather dataset. The primary goal was to leverage big data processing and machine learning techniques to create a scalable and efficient forecasting system to optimise solar energy management and grid integration.

The project's relevance and potential impact were highlighted, emphasising the importance of accurate solar energy forecasting for efficient operation and management of solar power systems. By utilising Spark's distributed computing capabilities and MLlib's machine learning algorithms, the project sought to build a robust and reliable model capable of handling large volumes of weather data in real-time.

The weather dataset, sourced from Kaggle, underwent an exploratory data analysis (EDA) to gain insights into its structure, summary statistics, data types, and missing values. The EDA revealed a large dataset with 1,710,802 rows and 18 columns, containing various weather-related variables such as temperature, dewpoint, precipitation, cloud cover, wind speed, and solar radiation. The dataset had no missing values, which was a positive analysis aspect.

Several visualisations were proposed to help understand the dataset better, including histograms or density plots to examine the distributions of variables and scatter plots,

correlation matrices, or pair plots to identify potential correlations or patterns among the variables.

The project addressed challenges such as handling large data volumes, feature selection, and model robustness. Spark's distributed processing capabilities were leveraged to efficiently load and process the historical weather dataset using Spark DataFrames. Feature scaling, encoding of categorical variables, and feature engineering techniques were applied to pre-process the data for machine learning tasks. The project employed techniques like splitting the data into training and testing sets, evaluating model performance using metrics like Root Mean Squared Error (RMSE), and hyperparameter tuning using cross-validation to ensure model robustness and reliability.

However, the project acknowledged PySpark's limitations in spatial autocorrelation analysis and structural equation modelling (SEM). PySpark lacks native support for spatial data types and spatial autoregressive models, making it challenging to conduct comprehensive spatial autocorrelation analysis without extensive user-defined functions (UDFs) or integration with external tools. Similarly, PySpark does not provide built-in support for SEM, requiring potential workarounds like exporting the data to a Pandas DataFrame and using specialised Python libraries.

Given the project's focus on utilising Spark MLlib and the challenges associated with integrating these techniques seamlessly into the PySpark workflow, spatial autocorrelation analysis and SEM were not pursued to maintain the project's focus on leveraging Spark MLlib for machine learning tasks.

The deployment step was also consciously excluded from the project's scope, as it required additional considerations and infrastructure setup that fell outside the primary goals of utilising Spark MLlib for data processing, feature engineering, model training, and evaluation. However, the importance of deployment in real-world scenarios was acknowledged, highlighting the need for careful planning, infrastructure setup, and consideration of factors such as scalability, security, and maintenance.

In conclusion, this project successfully demonstrated using Apache Spark's MLlib library and a comprehensive weather dataset to develop a machine learning model for short-term solar energy forecasting. The project effectively addressed challenges such as handling large data

volumes, feature selection, and model robustness within the PySpark framework. It showcased the application of Spark's distributed processing capabilities, feature engineering techniques, and model evaluation and tuning methods to build robust and reliable machine learning models. The project's findings and insights can enhance weather and climate modelling, forecasting, and downstream applications in various industries. The developed models and techniques can aid in optimising solar energy management, grid integration, and decision-making processes in the energy sector.

However, it is essential to acknowledge the limitations and challenges encountered during the project, such as PySpark's lack of native support for spatial autocorrelation analysis and SEM. These limitations highlight the need for further research and development of specialised tools and libraries that seamlessly integrate with PySpark to enable more comprehensive spatial and statistical analysis.

Moreover, excluding the deployment step from the project's scope underscores the importance of considering the operational aspects of deploying machine learning solutions in real-world scenarios. Future work could explore the deployment process, including model serialisation, containerisation, orchestration, API development, monitoring, and scalability considerations.

Overall, this project is a valuable contribution to solar energy forecasting and demonstrates the potential of leveraging big data processing and machine learning techniques using Apache Spark's MLlib library. The project's methodology, findings, and insights can be extended and applied to similar domains and problem statements, driving further advancements in the application of machine learning for renewable energy management and optimisation.

References

1. Chinthalapati, V. L. R. (2024). Big Data Analysis. MSc. in Data Science and Artificial Intelligence, Goldsmiths University of London.

2. Stamate, D. (2024). Machine Learning. MSc. in Data Science and Artificial Intelligence, Goldsmiths University of London.

3. Python Documentation. (2024). Retrieved from https://www.python.org/doc/

4. PySpark Documentation. (2024). Retrieved from https://spark.apache.org/docs/latest/api/python/index.html

5. Pandas Documentation. (2024). Retrieved from https://pandas.pydata.org/docs/#:~:text=,Mailing%20List

6. NumPy Documentation. (2024). Retrieved from https://numpy.org/doc/#:~:text=,User%20Guide%20PDF

7. Matplotlib Pyplot Documentation. (2024). Retrieved from https://matplotlib.org/stable/api/pyplot_summary.html#:~:text=,cases%20of%20programm

8. Seaborn Documentation. (2024). Retrieved from https://seaborn.pydata.org/#:~:text=,introductory%20notes%20or%20the%20paper