ASSIGNMENT 2

FORECASTING THE WIND POWER PRODUCTION IN ORKNEY

Github link: https://github.itu.dk/Pety/Power-Prediction/blob/main/Power_wind_project.ipynb

1. Introduction

In this report I will explain all the steps I have made for finding an optimal model that can predict the power generation in Orkey based on the wind direction and its speed. Moreover, mlflow package was used to track the performance of my models. Lastly, the best performing model was deployed by using Azure VM.

2. The Process

Here I will explain in detail all the steps I did to achieve power prediction. The stages can be divided into 4 main stages: 1) Fetching the data from database 2) Data preparation and feature engineering 3) Model training 4) Choosing and deploying the most optimal model.

2.1 The Data

Step 1. Fetch the data

The very first step was to fetch the data from two sources:

1. Orkney's renewable power generation - Sourced from Scottish and Souther Electricity Networks (SSEN)

2. Weather forecasts for Orkney- Sourced from the UK MetOffice

Since I had to fetch the data from two different places, I had 1 dataset with the power generation and another dataset regarding the wind direction. The power data was generated every minute and has the following columns: ANM,Non-ANM as shown on Figure1. The wind direction data was generated every three hours and has the following columns: Direction, Lead Hours, Source Time and Speed, as shown on Figure2.

time	ANM	Non-ANM	Total
2024-01-23 20:18:00+00:00	7.789766	12.200	19.989766
2024-01-23 20:19:00+00:00	7.646659	11.641	19.287659

Figure 1 Power Data Frame

	time	Direction	Lead_hours	Source_time	Speed
0	2024-01-23 21:00:00+00:00	SW	1	1706036400	11.17600
1	2024-01-24 00:00:00+00:00	WSW	1	1706047200	13.85824

Figure 2 Wind Data frame

Step 2: Encode the direction of the wind

I encoded wind direction using degrees for the 16 directions in the dataset. This method follows the tradition of using degrees for direction, as seen on a compass card. I encoded wind direction in degrees to make it easier for machine learning models to understand and analyse the data. This method aligns with traditional navigation practices, where numerical values represent direction, such as "North" corresponding to 0 degrees and "Northwest" to 315 degrees.

Step 3: Merge both datasets and fill in missing values

The next step was to merge the datasets by using a left join on the power data. This way I kept most of the data, which is valuable for making a model. However, this method created a lot of missing values.

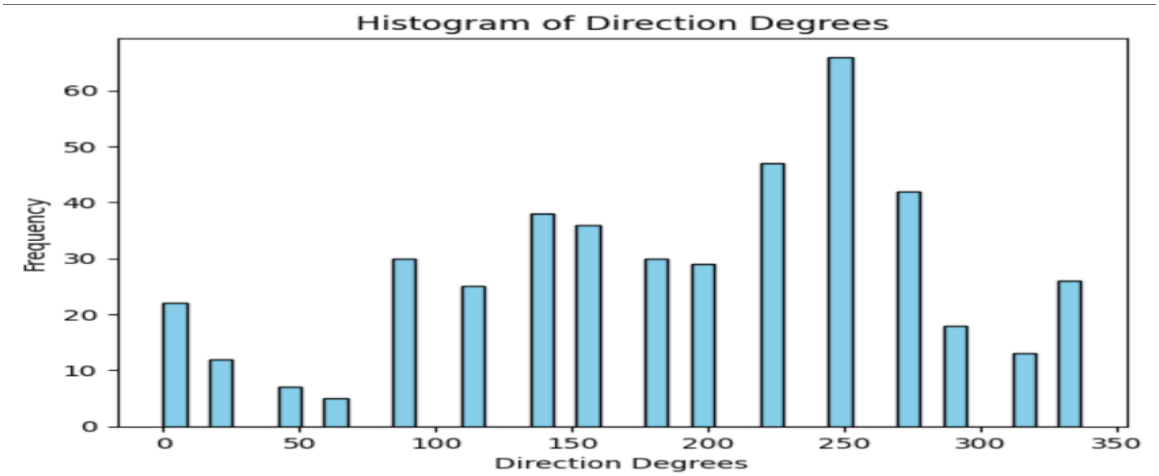
```
merged = pd.merge(power_df, wind_df, on = "time", how='left')
```

To fix the problem, I used interpolation, specifically 'bfill'. This method fills in the missing values by looking at the data points that come after them and using those values to fill the gaps. This way, I ensured the dataset was more complete for analysis.

```
newdf = merged.interpolate(method='bfill')
newdf.head()
```

✓ 0.0s Python

After using that method, I have decided to plot a histogram for the direction degrees, just to see how the positions are distributed and if the bfill method made sense to use, and it looked reasonable.



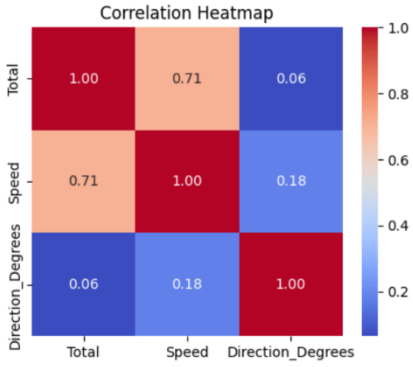
Step 4: Selecting the features

I have decided to use the Speed and the Direction of the wind as independent variables that will predict power; therefore, I removed the following columns from the merged dataset.

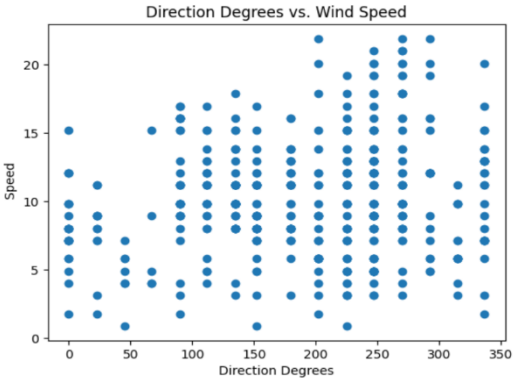
```
# Dropping the non needed columns

columns_to_remove = ['time','ANM','Non-ANM', 'Lead_hours', 'Source_time']
merged.drop(columns=columns_to_remove, inplace=True)
```

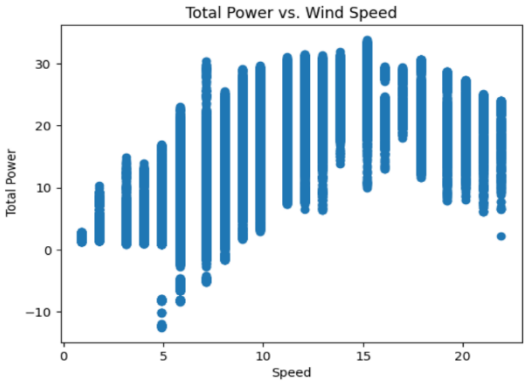
✓ 0.2s Python

In order to see the relations between all the variables, I have decided to calculate their correlations and plot them. As seen on the figure, there is a high corelation between the Speed of the wind and the generated power, however the direction of the wind does not have a strong correlation.



I have investigated that further by plotting the total power produced vs the speed and the direction of the wind. There is certainly no linear relationship between these features and feature scaling had to be done as well.





2.2 Model Selection

Predicting the power based on the wind direction and the speed of the wind, is a regression problem so I had to come up with regression classifiers. I have decided to compare the performance of the following models: Linear Regression using only the Speed, Linear Regression using only the direction of the wind, Linear Regression using both features, kNN and Random Forest using both features. On the Figure below, it can be seen the comparison of the Mean Squared Error for each model by using the mlflow package.

The reason for choosing the Mean Absolute Error as the metric to compare the performance of different regression models is because it gives a clear picture of how well each model predicts power based on wind speed and direction. It helps us see which model makes the most accurate predictions overall.

Step 5. Splitting the data

Firstly, I had split the data into training (60%), development (20%) and testing data (20%).

```
# Split the data into train, dev, and test sets
X_train_dev, X_test, y_train_dev, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

X_train, X_dev, y_train, y_dev = train_test_split(
    X_train_dev, y_train_dev, test_size=0.25, random_state=42
)
```

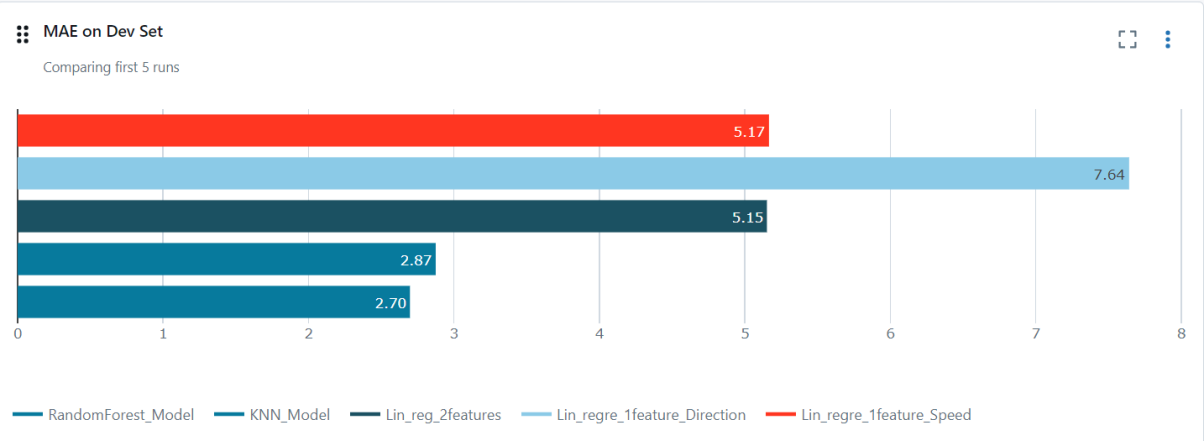
Step 5. Feature Scaling

Feature scaling makes sure that all our features are on a similar scale, so algorithms can understand them better. It's especially important when some features, like wind direction, have big numbers, so everything is treated equally.

```
# Select our feature variables and our target variable.
X = df_cleaned["Direction_Degrees"].values.reshape(-1,1)
y = df_cleaned["Total"].values.reshape(-1,1)
```

Step 6. Models performance on dev data

I have used the development data to test the performance of all the models and concluded that Random Forest and kNN have similar results, but Random Forests is giving slightly better results. So I have decided to further tune that model.



Step 7. Find the best parameters for the Random Forest model using Grid Search.

The parameters could be seen down on the Figure.

```
# Define the best parameters found during grid search
best_params = {
    'n_estimators': 100,    'max_depth': 10,    'min_samples_split': 2,
    'min_samples_leaf': 1,
    'max_features': 'auto'
}
```

Step 8. Test the model on the test data

Here are the results, after testing the tuned model on the test data.

```
MAE on Test Set: 2.6809233604679674
```

2.3 Model Deployment

Due to Conda issues, the local deployment of my model was unsuccessful, therefore the Azure VM deployment was also unsuccessful.