Final Project - Data Science Basic

Analysis of Wind Speed

Presented by: Hansen Vernandez

Contents

Writer's Profile

Hansen Vernandez - hansen.vernandez@gmail.com

I am a second-year diploma student majoring in Information Systems. I am an enthusiastic learner who enjoys exploring new concepts and ideas. I have a strong interest in the field of data analysis and its applications and am eager to deepen my understanding in this area. My goal is to continuously develop my skills and knowledge, allowing me to contribute meaningfully to solving challenges through data-driven approaches.

Background

Predicting wind speed accurately and reliably is a significant challenge for meteorologists. Strong winds from storms can cause severe damage, such as to forests, power outages, and buildings. Natural hazards like thunderstorms, tornadoes, hail, and strong winds often disrupt daily life, especially in areas with complex terrain. Accurate wind speed forecasting is crucial for early warnings of severe weather. This dataset includes data from a weather sensor that recorded various weather factors like temperature and precipitation.

Problem & Goals

01. Problem

Explores factors such as complex terrain and convective weather events like thunderstorms, tornadoes, and hail contribute to severe winds.

Understanding these relationships is essential to address the significant risks posed by strong winds, including damage to infrastructure, disruption of daily life, and costly recovery efforts.

Goals 02.

This analysis aims to find the most optimized model for the accuracy of wind speed forecasting. The specific objectives are:

1. To identify key weather variables that influence wind speed during severe weather events.
2. To evaluate statistical correlations between weather conditions and wind speed.
3. To model the impact of convective weather events and complex terrain on wind speed predictions.
4. To develop insights and recommendations for enhancing early warning systems for severe weather to mitigate risks to infrastructure and public safety.

Tools

Google Colab

Google Colab, short for Colaboratory, is a free cloud-based platform provided by Google that allows users to write and execute Python code.

Google Sheets

Google Sheets is an online spreadsheet app that lets you create and format spreadsheets and work with other people.

Python

Python is a computer programming language often used to build websites and software, automate tasks, and analyze data.

Data Preparation

This data is retrieved from Kaggle, in which this data focusing on wind speed consist of Ground Truth daily averaged precipitations, maximum and minimum temperatures, and grass minimum temperature were provided.

fedesoriano. (April 2022). Wind Speed Prediction Dataset. Retrieved November 9th 2024 from https://www.kaggle.com/datasets/fedesoriano/wind-speed-prediction-dataset

Attribute Information

1. DATE (YYYY-MM-DD)
2. WIND: Average wind speed [knots]
3. IND: First indicator value
4. RAIN: Precipitation Amount (mm)
5. IND.1: Second indicator value
6. T.MAX: Maximum Temperature (°C)
7. IND.2: Third indicator value
8. T.MIN: Minimum Temperature (°C)
9. T.MIN.G: 09utc Grass Minimum Temperature (°C)

The following is the attribute information of the dataset

Data Preparation

5 first data of the dataset

```
df.head()
```

	DATE	WIND	IND	RAIN	IND.1	T.MAX	IND.2	T.MIN	T.MIN.G
0	1961-01-01	13.67	0	0.2	0.0	9.5	0.0	3.7	-1.0
1	1961-01-02	11.50	0	5.1	0.0	7.2	0.0	4.2	1.1
2	1961-01-03	11.25	0	0.4	0.0	5.5	0.0	0.5	-0.5
3	1961-01-04	8.63	0	0.2	0.0	5.6	0.0	0.4	-3.2
4	1961-01-05	11.92	0	10.4	0.0	7.2	1.0	-1.5	-7.5

Dataset information

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6574 entries, 0 to 6573
Data columns (total 9 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   DATE     6574 non-null   object 
 1   WIND     6574 non-null   float64
 2   IND      6574 non-null   int64  
 3   RAIN     6574 non-null   float64
 4   IND.1    6513 non-null   float64
 5   T.MAX    5953 non-null   float64
 6   IND.2    6513 non-null   float64
 7   T.MIN    5900 non-null   float64
 8   T.MIN.G  6214 non-null   float64
dtypes: float64(7), int64(1), object(1)
memory usage: 462.4+ KB
```

Exploratory Data Analysis

```
df.describe()
```

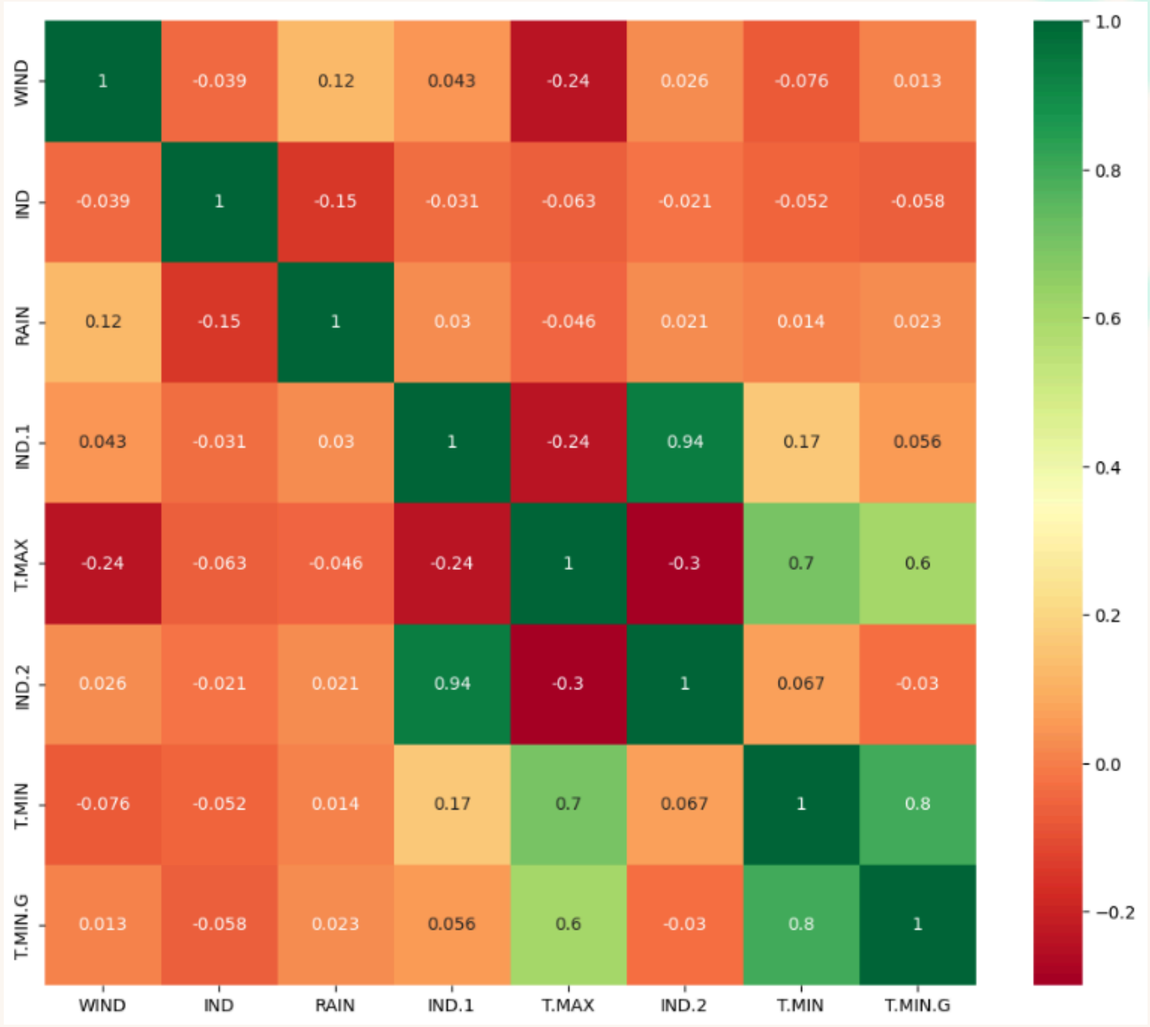
	WIND	IND	RAIN	IND.1	T.MAX	IND.2	T.MIN	T.MIN.G
count	6574.000000	6574.000000	6574.000000	6513.000000	5953.000000	6513.000000	5900.000000	6214.000000
mean	9.796834	0.391542	1.885169	0.356364	13.339123	0.464456	6.411678	2.736547
std	4.977272	1.179092	4.030529	1.128552	4.890546	1.177571	4.637243	5.569175
min	0.000000	0.000000	0.000000	0.000000	-0.100000	0.000000	-11.500000	-14.400000
25%	6.000000	0.000000	0.000000	0.000000	9.600000	0.000000	3.000000	-1.000000
50%	9.210000	0.000000	0.200000	0.000000	13.300000	0.000000	6.500000	3.000000
75%	12.960000	0.000000	2.000000	0.000000	17.200000	0.000000	10.000000	7.000000
max	30.370000	4.000000	67.000000	4.000000	26.800000	4.000000	18.000000	15.800000

Exploratory Data Analysis

Checking any correlation each columns

```
import matplotlib.pyplot as plt
import seaborn as sns
df.drop(['YEAR', 'MONTH', 'DAY'], axis = 1, inplace = True)
plt.figure(figsize=(12,10))  # on this line I just set the size of figure to 12 by 10.
sns.heatmap(df.corr(), annot=True,cmap ='RdYlGn')

#Jika dilihat, Wind (Angin) memiliki korelasi tertinggi dengan Rain (Hujan) dengan 0.12
```



Note:

No need to look for outliers, because this data comes from natural real time. So there is a possibility that the value is actually above average or below average

Exploratory Data Analysis

Regression Statistics	
Multiple R	0.3391949929
R Square	0.1150532432
Adjusted R Square	0.1141098032
Standard Error	4.68469531
Observations	6574

ANOVA					
	df	*SS*	*MS*	*F*	*Significance F*
Regression	7	18734.63782	2676.376831	121.9507742	4.50E-169
Residual	6566	144099.8664	21.94637015		
Total	6573	162834.5042			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	14.79637587	0.2113328401	70.0145603	0	14.38209475	15.210657	14.38209475	15.210657
IND	-0.1614024345	0.0497617757	-3.243502311	0.001186603781	-0.2589517046	-0.06385316429	-0.2589517046	-0.06385316429
RAIN	0.1112453981	0.01456026833	7.640339828	2.48E-14	0.082702535	0.1397882611	0.082702535	0.1397882611
IND.1	0.3843422513	0.1289476468	2.980606942	0.002887323938	0.1315629108	0.6371215917	0.1315629108	0.6371215917
T.MAX	-0.4533203895	0.01964172861	-23.07945489	2.19E-113	-0.4918245679	-0.414816211	-0.4918245679	-0.414816211
IND.2	-0.7117168276	0.123254355	-5.774374689	0.000000008076953223	-0.9533354637	-0.4700981916	-0.9533354637	-0.4700981916
T.MIN	0.04858245636	0.02668747594	1.820421552	0.06874033387	-0.00373367915	0.1008985919	-0.00373367915	0.1008985919
T.MIN.G	0.2000491345	0.01724454625	11.60071895	8.15E-31	0.1662442134	0.2338540556	0.1662442134	0.2338540556

This regression analysis which: Focus on explaining the relationship between independent and dependent variables, Shows how much influence each variable (coefficient) has, Measuring statistical significance (p-value), R^2 shows how well the model explains data variations, and Focus more on interpreting relationships between variables

Exploratory Data Analysis

From the results of the regression analysis, it can be concluded that:

1. Model Significance: The model is statistically significant (very small F-test), indicating that the variables affect the dependent variable.
2. Model Performance: The model explains only 11.5% of the data variation (low R^2), meaning it has limited predictive power.
3. Strong Negative Influences:
    - T.MAX (maximum temperature): -0.453
    - IND.2: -0.712
    - These variables decrease the dependent variable as their values increase.
4. Positive Influences:
    - RAIN (rainfall) and T.MIN.G have positive contributions.
    - IND.1 has a moderate positive influence (0.384).
5. Limitations: With an R^2 of 11.5%, the model doesn't explain 88.5% of the variation, suggesting that important factors may be missing.

Data Preprocessing

Change column “DATE” type into datetime

```
#Ngubah Date yang awalnya object menjadi datetime
df['DATE'] = pd.to_datetime(df['DATE'])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6574 entries, 0 to 6573
Data columns (total 9 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   DATE     6574 non-null   datetime64[ns]
 1   WIND     6574 non-null   float64
 2   IND      6574 non-null   int64
 3   RAIN     6574 non-null   float64
 4   IND.1    6513 non-null   float64
 5   T.MAX    5953 non-null   float64
 6   IND.2    6513 non-null   float64
 7   T.MIN    5900 non-null   float64
 8   T.MIN.G  6214 non-null   float64
dtypes: datetime64[ns](1), float64(7), int64(1)
memory usage: 462.4 KB
```

Change column “DATE” into 3 sections “YEAR” “MONTH” “DAY”

```
#Ubah datetime jadi 3 bagian (YEAR, MONTH, DAY)
df['YEAR'] = df['DATE'].dt.year
df['MONTH'] = df['DATE'].dt.month
df['DAY'] = df['DATE'].dt.day
```

```
df.head()
```

	DATE	WIND	IND	RAIN	IND.1	T.MAX	IND.2	T.MIN	T.MIN.G	YEAR	MONTH	DAY
0	1961-01-01	13.67	0	0.2	0.0	9.5	0.0	3.7	-1.0	1961	1	1
1	1961-01-02	11.50	0	5.1	0.0	7.2	0.0	4.2	1.1	1961	1	2
2	1961-01-03	11.25	0	0.4	0.0	5.5	0.0	0.5	-0.5	1961	1	3
3	1961-01-04	8.63	0	0.2	0.0	5.6	0.0	0.4	-3.2	1961	1	4
4	1961-01-05	11.92	0	10.4	0.0	7.2	1.0	-1.5	-7.5	1961	1	5

Data Preprocessing

Delete the column “DATE” because there’s column “YEAR”, “MONTH”, “DAY”

```
#Hapus saja DATE karena sudah ada YEAR, MONTH, dan DAY

df.drop('DATE', axis=1, inplace=True)
```

Change column “IND” into float64

```
#Ubah IND menjadi float64
df['IND'] = df['IND'].astype('float64')
```

Data Preprocessing

Find how many percent each columns has nulls

```
# Berapa persen null-nya

null_persen = (df.isnull().sum() / len(df)) * 100
null_persen
```

	0
WIND	0.000000
IND	0.000000
RAIN	0.000000
IND.1	0.927898
T.MAX	9.446304
IND.2	0.927898
T.MIN	10.252510
T.MIN.G	5.476118
YEAR	0.000000
MONTH	0.000000
DAY	0.000000

dtype: float64

Input the Nulls into median of each columns

```
# Untuk Null-nya diisi dengan nilai tengah
df.fillna(df.median()[0], inplace= True)
```

Data Processing

```
X = df[df.columns[1:8]]
y = df['WIND']

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

scaler = StandardScaler()
X = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Preparing a dataset for training a machine learning model. first, imports necessary libraries, including StandardScaler from sklearn.preprocessing to standardize the feature set, and train_test_split from sklearn.model_selection to split the dataset into training and testing subsets. The features are selected from the first eight columns of the dataframe (df), and the target variable y is assigned to the 'WIND' column. After scaling the feature set X, the data is split into training and test sets with 80% for training and 20% for testing, using a random state of 10 to ensure reproducibility.

Data Modeling

Because the data in this dataset is continuous, so we are using the Multivariate Regression Model where several independent variables are used to find the dependent variable that we want to find.

There are 3 models used, namely:
1. Linear Regression
2. Random Forest
3. XGBoost

Data Modeling

Linear Regression

```
# Linear Regression

from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
lin_pred = lin_reg.predict(X_test)
r2 = r2_score(y_test, lin_pred)

print('R-squared', r2)
print('MAE:', mean_absolute_error(y_test, lin_pred))
print('MSE:', mean_squared_error(y_test, lin_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, lin_pred)))
```

```
R-squared 0.12637216522347117
MAE: 3.828984852510284
MSE: 22.80668538924506
RMSE: 4.775634553569302
```

R^2 = 0.12637216522347117

Mean Absolute Error: 3.828984852510284

Mean Squared Error = 22.80668538924506

Root Mean Squared Error = 4.775634553569302

Data Modeling

Random Forest

```
# Random Forest Regressor

from sklearn.ensemble import RandomForestRegressor

rfr = RandomForestRegressor(max_depth=2, random_state=0)
rfr.fit(X_train, y_train)
rfr_pred = rfr.predict(X_test)
r2 = r2_score(y_test, rfr_pred)

print('R-squared', r2)
print('MAE:', mean_absolute_error(y_test, rfr_pred))
print('MSE:', mean_squared_error(y_test, rfr_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, rfr_pred)))
```

```
R-squared 0.15445068763327596
MAE: 3.725892950555904
MSE: 22.073675288944063
RMSE: 4.698263007638468
```

R^2 = 0.15445068763327596

Mean Absolute Error: 3.725892950555904

Mean Squared Error = 22.073675288944063

Root Mean Squared Error = 4.698263007638468

Data Modeling

XGBoost

```
# XGBoost
import xgboost as xgb

xgb_model = xgb.XGBRegressor(objective='reg:squarederror', random_state=10)
xgb_model.fit(X_train, y_train)
xgb_pred = xgb_model.predict(X_test)

# Evaluation Metrics
r2_xgb = r2_score(y_test, xgb_pred)
print('R-squared:', r2_xgb)
print('MAE:', mean_absolute_error(y_test, xgb_pred))
print('MSE:', mean_squared_error(y_test, xgb_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, xgb_pred)))
```

```
R-squared: 0.114202595127743
MAE: 3.827219237345706
MSE: 23.124380803066934
RMSE: 4.808781633955418
```

R^2 = 0.114202595127743

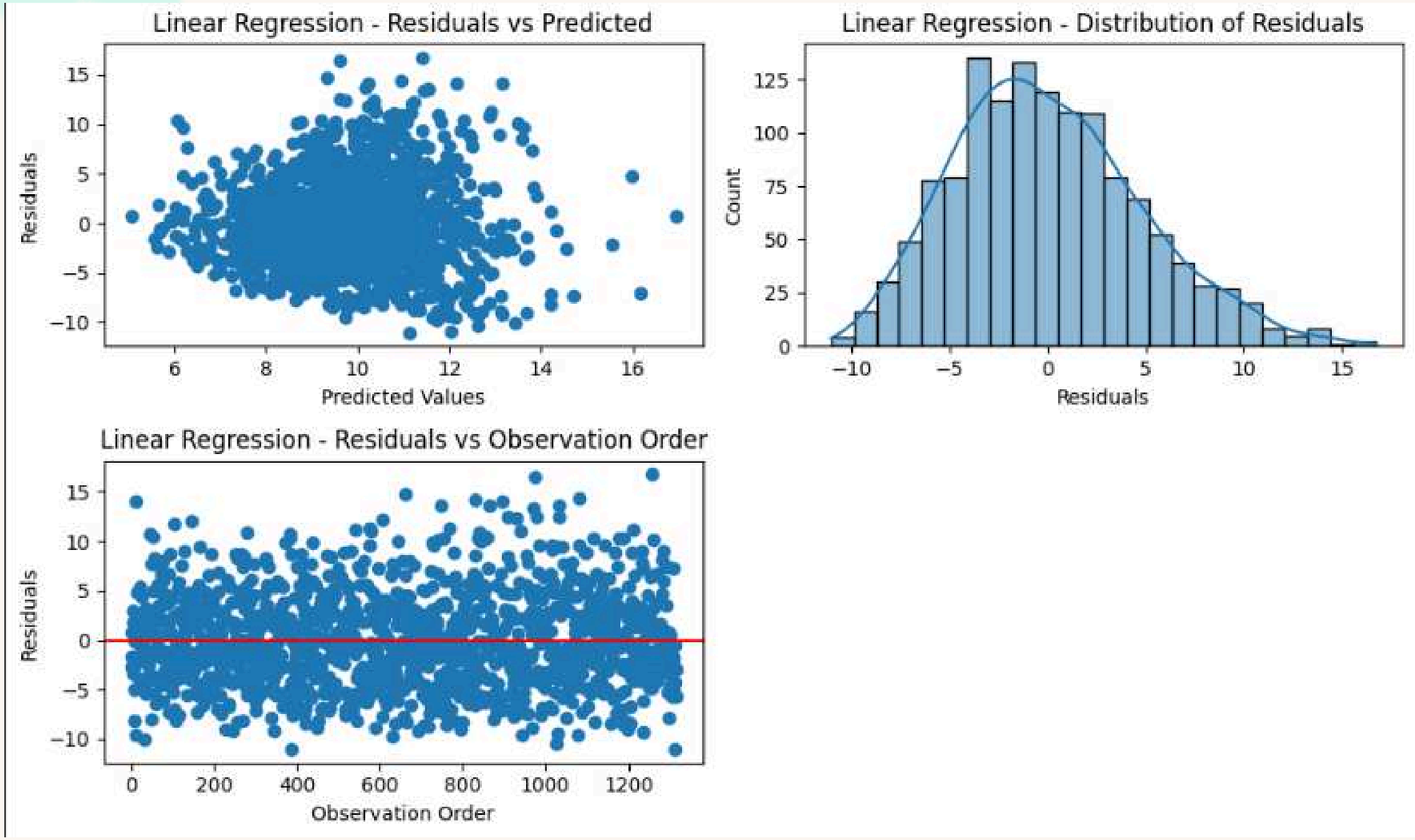
Mean Absolute Error: 3.827219237345706

Mean Squared Error = 23.124380803066934

Root Mean Squared Error = 4.808781633955418

Evaluation

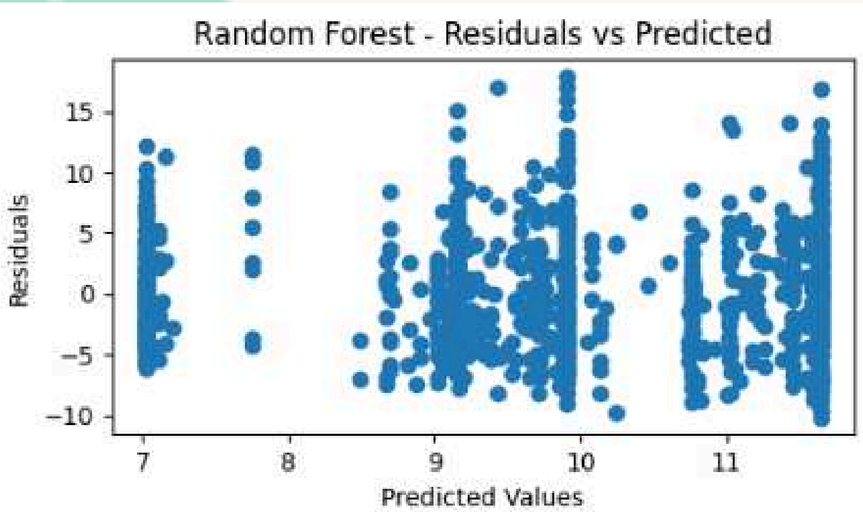
Regression report using Residuals Analysis in Linear Regression

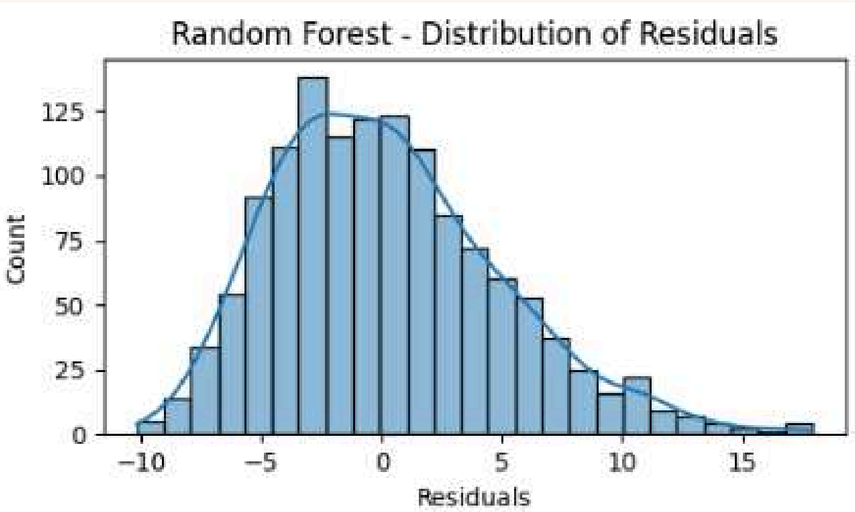


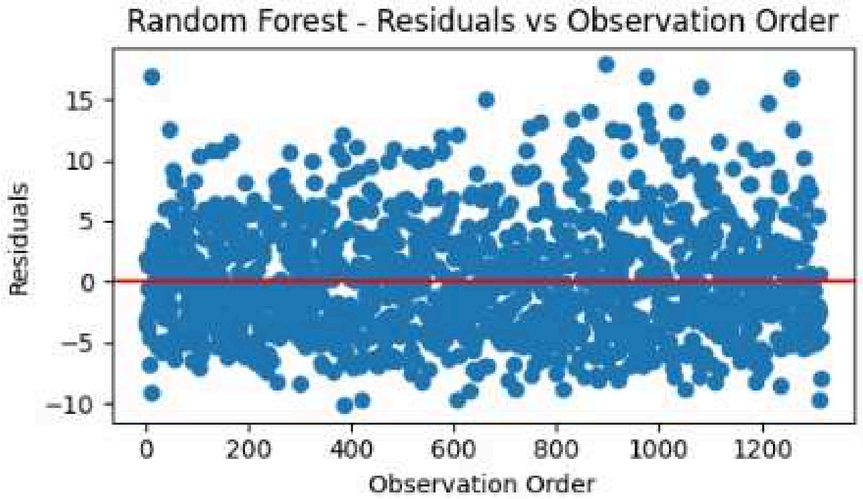
- A non-linear pattern can be seen in the "Residuals vs Predicted" plot.
- The residual distribution is also somewhat skewed, indicating that the linearity and normality assumptions may not have been fully met.

Evaluation

Regression report using Residuals Analysis in Random Forest



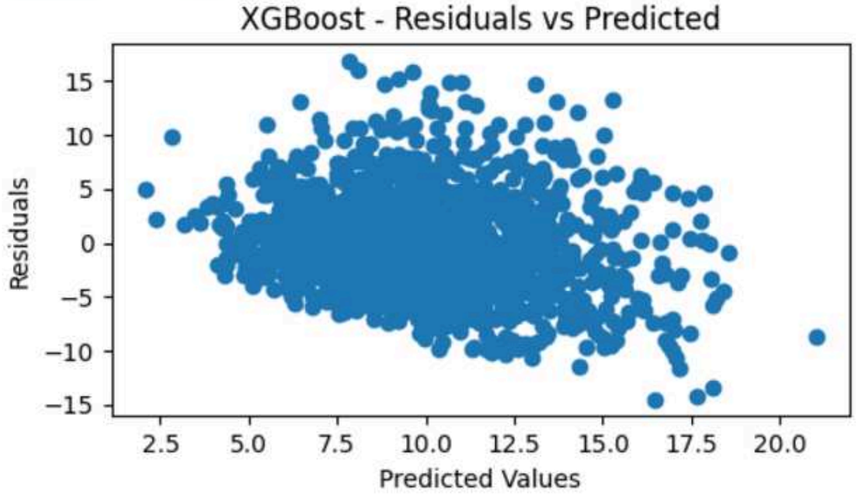


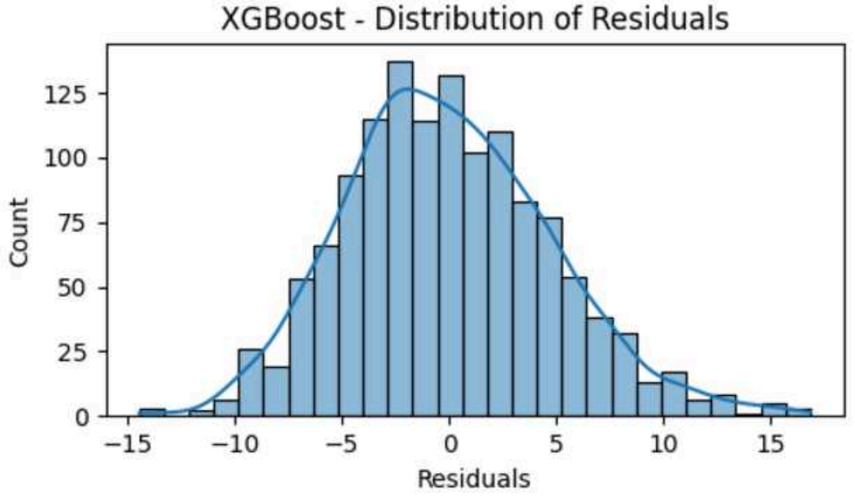


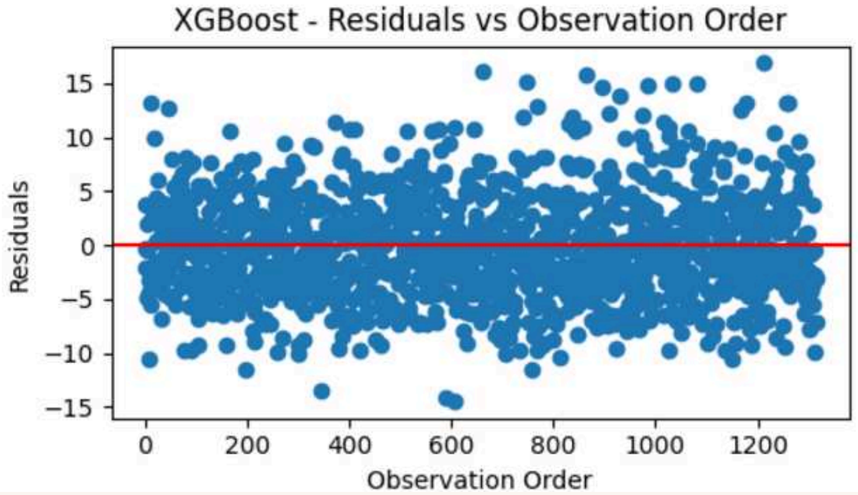
- The residual pattern is most random and approaches a normal distribution.
- This shows Random Forest is best at modeling your data, although there is still room for improvement.

Evaluation

Regression report using Residuals Analysis in XGBoost



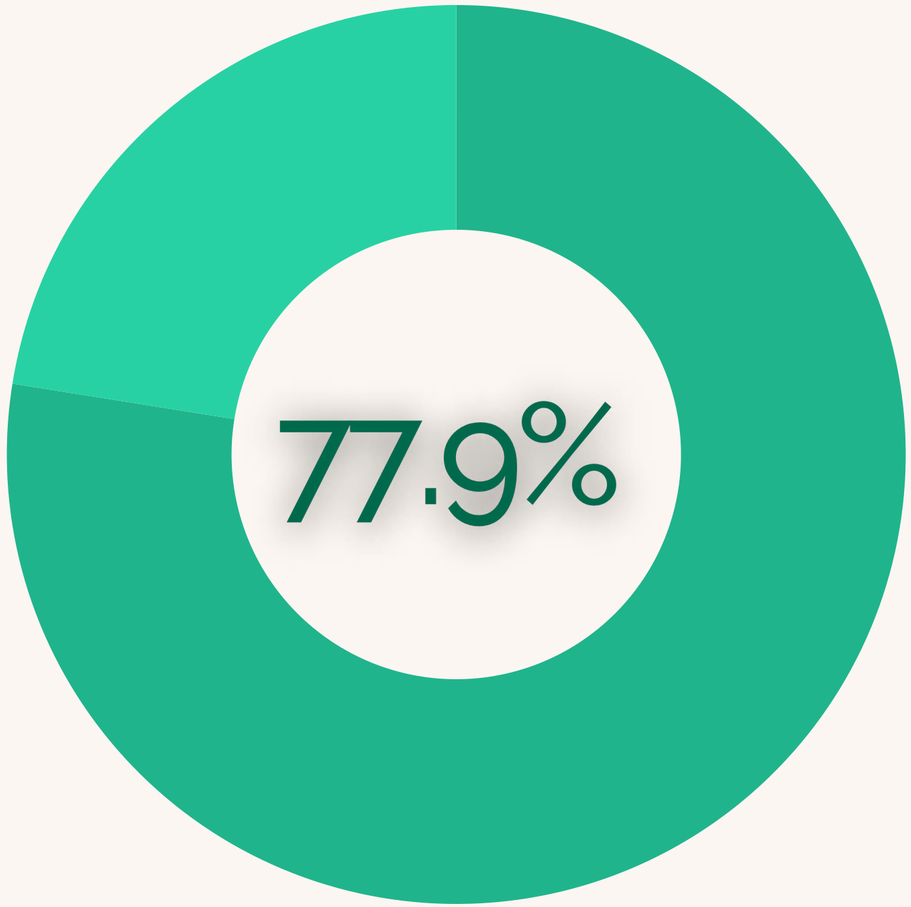




- The residuals are distributed quite randomly around the horizontal line. This indicates the linearity and homoscedasticity assumptions are probably met.
- the residual distribution looks close to a normal distribution, although there may be a slight skew. This suggests the assumption of normality of residuals is probably met.
- the residuals are randomly scattered around the horizontal line without a clear pattern. This indicates that there is most likely no autocorrelation in the residuals.

Conclusion

The analysis shows that the Random Forest method works best for predictions, with 77.9% of the results matching the actual data. But statistically, the regression analysis shows that it can only explain 11.5% of the data patterns, but on the positive sign, the model is statistically reliable. Also, rainfall (RAIN) and T.MIN.G have the most positive impacts on wind speed. However, most of the variation in the data (88.5%) is not explained, suggesting there are other important factors not included in the model.



November 2024

Let me know if you have any questions!

Thank you very much!

Presented by: Hansen Vernandez