Weather Prediction Using Classification Model

Project Overview

Weather forecasting plays a significant role in different aspects of life such as in the operation of hydro-power plants, renewable energy, flood management, and agriculture. Prediction of weather phenomena is of major interest for human society to avoid or minimize the destruction of weather hazards. Numerous efforts were made to make weather prediction as accurate as possible. Recently, machine learning techniques have been used for weather forecasting for large periods of time, as it is more accurate than models based on physical principles. To address various problems, varieties of machine learning algorithms are applied in different fields. In this project, to examine how much accurate these models to predict weather conditions. It is carried out to compare these models so that it can be relized which model works better. In this project, a set of the most common machine learning techniques are explored to generate robust weather forecasting model for long periods of time. Moreover, the combinations of all the model parameters are considered for simulations. The experimental results of the classifiers show that which classifier model gives better classification accuracy. In this project numpy, pandas, matplotlib,seaborn, sklearn libries are used. Data preprocessing and Exploretory data analysis is present to justify the project as more accurate one. After data analysis unnecessary variables were removed. Then dataset is splited into traning and test set. After that a model is build for each model and test our data. At last it is compared the accuracy score.

Dataset Overview

We have used a dataset about weather information from kaggle to train our model and evaluate by testing.

Number of Instances: 1462

Number of Attributes: 6 numeric predictive.

Attribute Information (in order):

date: YYYY-MM-DD

precipitation: All forms in which water falls on the land surface and open water bodies as rain, sleet, snow, hail, or drizzle

temp_max: Maximum Temperature

temp_min: Minimum Temperature

wind: Wind speed

weather: target

Using the Columns :

- date
- precipitation
- tempmax
- tempmin
- wind

We are going to predict the weather condition :

- drizzle
- rain
- sun
- snow
- fog

Missing Attribute Values: None

Creators: Bala Vashan, ANANTH R, Hrishikesh Yadav

This dataset is taken from Kaggle.com. The name of the dataset is Weather prediction. The Dataset contains data for 4 years (from 1st January 2012 to 31st December 2015)

The art of weather prediction has been a difficult task for many of the researchers and analysts. Its very important to predict weather now a day. So we decided to predict weather using data set which is downloaded from kaggle to train our model and evaluate by testing using 6 attribute. We are going to predict 5 types of weather conditions (drizzle, rain, sun, snow, fog).

This is a copy of Weather prediction dataset:

Weather Prediction | Kaggle Dataset Link : [https://www.kaggle.com/datasets/ananthr1/weather-prediction (https://www.kaggle.com/datasets/ananthr1/weather-prediction)](https://www.kaggle.com/datasets/ananthr1/weather-prediction)

Import Libraries

In [1]:
```
# importing necessary libaries
import numpy as np
import pandas as pd

#visualizing libaries
import matplotlib.pyplot as plt
import seaborn as sns
```

Load Dataset

In [2]:
```
#importing dataset from a csv file
data = pd.read_csv("seattle-weather.csv")
data.head(10)
```

Out[2]:

	date	precipitation	temp_max	temp_min	wind	weather
0	2012-01-01	0.0	12.8	5.0	4.7	drizzle
1	2012-01-02	10.9	10.6	2.8	4.5	rain
2	2012-01-03	0.8	11.7	7.2	2.3	rain
3	2012-01-04	20.3	12.2	5.6	4.7	rain
4	2012-01-05	1.3	8.9	2.8	6.1	rain
5	2012-01-06	2.5	4.4	2.2	2.2	rain
6	2012-01-07	0.0	7.2	2.8	2.3	rain
7	2012-01-08	0.0	10.0	2.8	2.0	sun
8	2012-01-09	4.3	9.4	5.0	3.4	rain
9	2012-01-10	1.0	6.1	0.6	3.4	rain

In [3]:
```
data.tail(10)
```

Out[3]:

	date	precipitation	temp_max	temp_min	wind	weather
1451	2015-12-22	4.6	7.8	2.8	5.0	rain
1452	2015-12-23	6.1	5.0	2.8	7.6	rain
1453	2015-12-24	2.5	5.6	2.2	4.3	rain
1454	2015-12-25	5.8	5.0	2.2	1.5	rain
1455	2015-12-26	0.0	4.4	0.0	2.5	sun
1456	2015-12-27	8.6	4.4	1.7	2.9	rain
1457	2015-12-28	1.5	5.0	1.7	1.3	rain
1458	2015-12-29	0.0	7.2	0.6	2.6	fog
1459	2015-12-30	0.0	5.6	-1.0	3.4	sun
1460	2015-12-31	0.0	5.6	-2.1	3.5	sun

In [4]:
```
#information about our data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1461 entries, 0 to 1460
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   date           1461 non-null   object 
 1   precipitation  1461 non-null   float64
 2   temp_max       1461 non-null   float64
 3   temp_min       1461 non-null   float64
 4   wind           1461 non-null   float64
 5   weather        1461 non-null   object 
dtypes: float64(4), object(2)
memory usage: 68.6+ KB
```

Preprocessing

In [5]:
```
#Checking 0 value arries or not for the all attributes
data.isnull().sum()
```

Out[5]:
```
date             0
precipitation    0
temp_max         0
temp_min         0
wind             0
weather          0
dtype: int64
```

In [6]:
```
#convert date in datetime format
data['date'] = pd.to_datetime(data['date'])
```

In [7]:
```
#replace null value of date features
data['date'].replace(0, np.nan, inplace=True)
```

In [8]:
```
data.isnull().sum()
```

Out[8]:
```
date             0
precipitation    0
temp_max         0
temp_min         0
wind             0
weather          0
dtype: int64
```

In [9]:
```
data.isnull().sum()/len(data)
```

Out[9]:
```
date             0.0
precipitation    0.0
temp_max         0.0
temp_min         0.0
wind             0.0
weather          0.0
dtype: float64
```

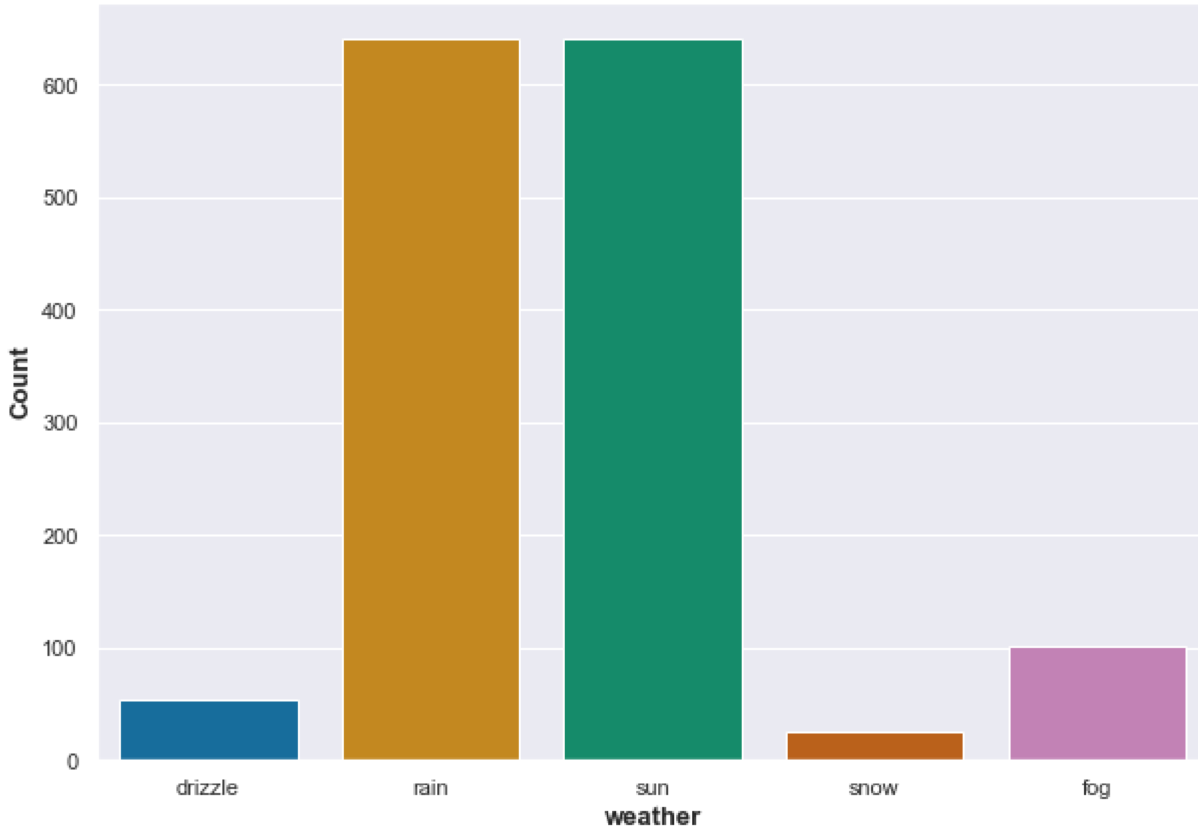
In [10]:
```
print(data.describe())
```

```
       precipitation     temp_max     temp_min         wind
count    1461.000000  1461.000000  1461.000000  1461.000000
mean        3.029432    16.439083     8.234771     3.241136
std         6.680194     7.349758     5.023004     1.437825
min         0.000000    -1.600000    -7.100000     0.400000
25%         0.000000    10.600000     4.400000     2.200000
50%         0.000000    15.600000     8.300000     3.000000
75%         2.800000    22.200000    12.200000     4.000000
max        55.900000    35.600000    18.300000     9.500000
```

Here we can see that no data is missing here. For precipitation,temp_max, temp_min & weather, it is not possible to hold a zero value. But for date & wind, it is not possible to hold a zero value. So we convert our date to datetime format and then check weather there is any zero value or not into date or wind. To check, we replace all 0's of date to NAN. Then check the 0 values.

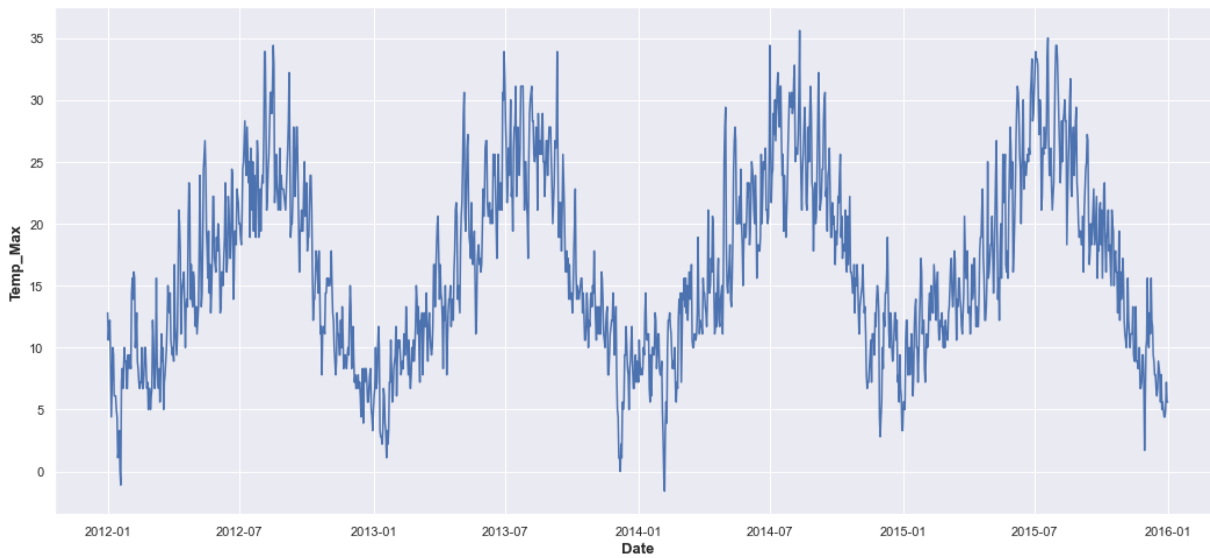
Exploratory data analysis

In [11]:
```
#Count plot against weather and data
plt.figure(figsize=(10,7))
sns.set_theme()
sns.countplot(x = 'weather',data = data,palette="colorblind")
plt.xlabel("weather",fontweight='bold',size=13)
plt.ylabel("Count",fontweight='bold',size=13)
plt.show()
```



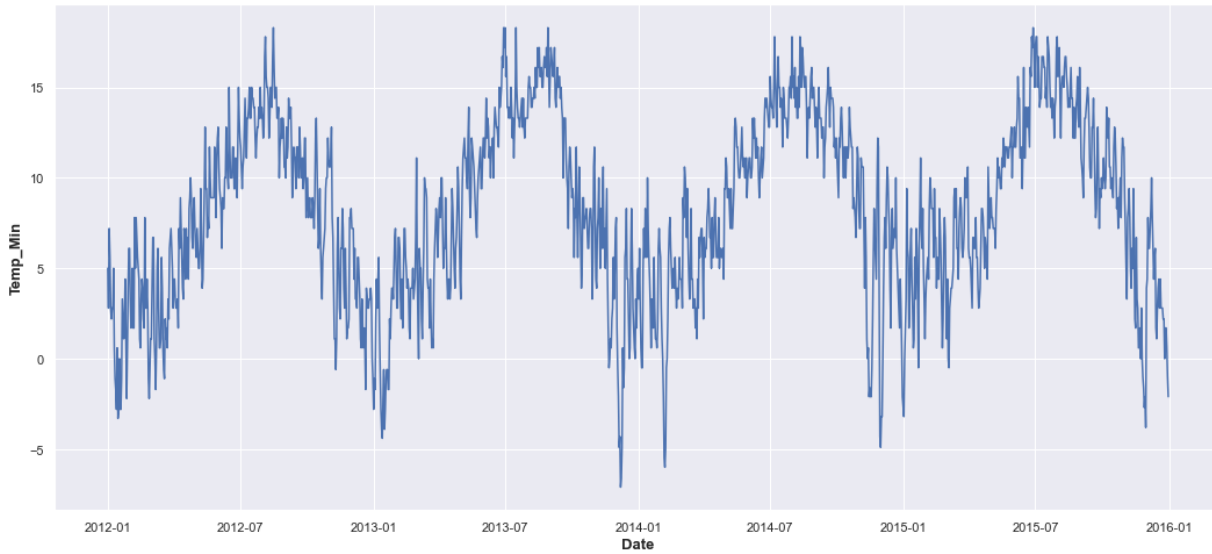
Here we count the value of target attributes value. We can see that most of the time weather are sunny and rainy above 600. But only few times weather is foggy around 100. Drizzly or foggy are a few in count below 100.

In [12]:
```
# Maximum temperature changing rate with respect to date
plt.figure(figsize=(18,8))
sns.set_theme()
sns.lineplot(x = 'date',y='temp_max',data=data)
plt.xlabel("Date",fontweight='bold',size=13)
plt.ylabel("Temp_Max",fontweight='bold',size=13)
plt.show()
```



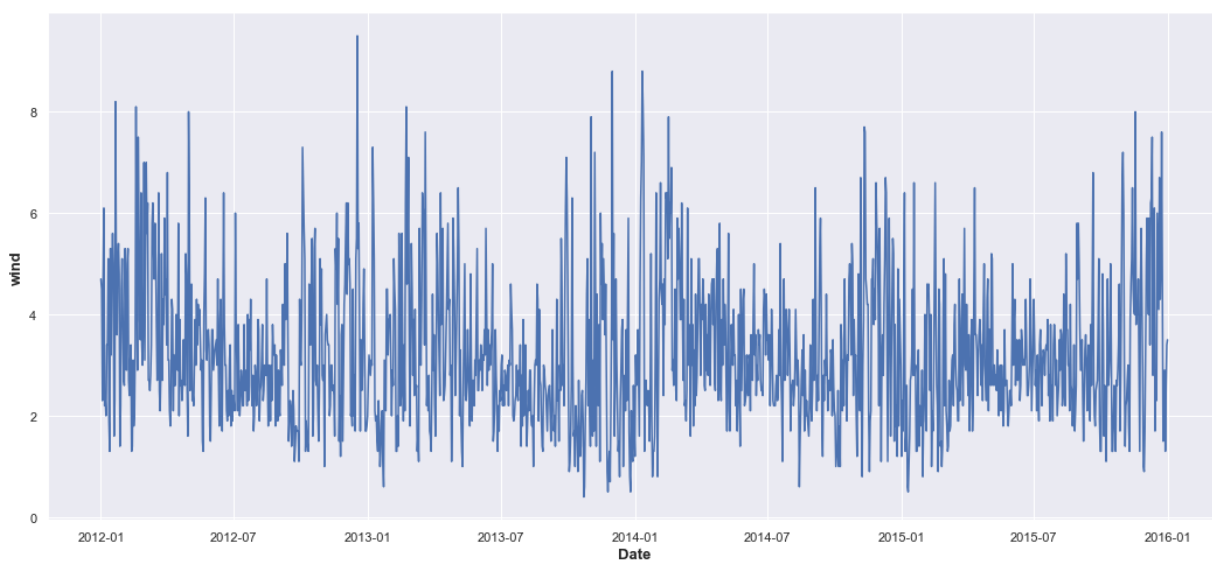
Here we can see form the line plot that maximum tempare increase around 7th month of each year.

In [13]:
```
# Minimum temperature changing rate with respect to date
plt.figure(figsize=(18,8))
sns.set_theme()
sns.lineplot(x = 'date',y='temp_min',data=data)
plt.xlabel("Date",fontweight='bold',size=13)
plt.ylabel("Temp_Min",fontweight='bold',size=13)
plt.show()
```



Here we can see form the line plot that minimum temparature also increase around 7th month of each year.

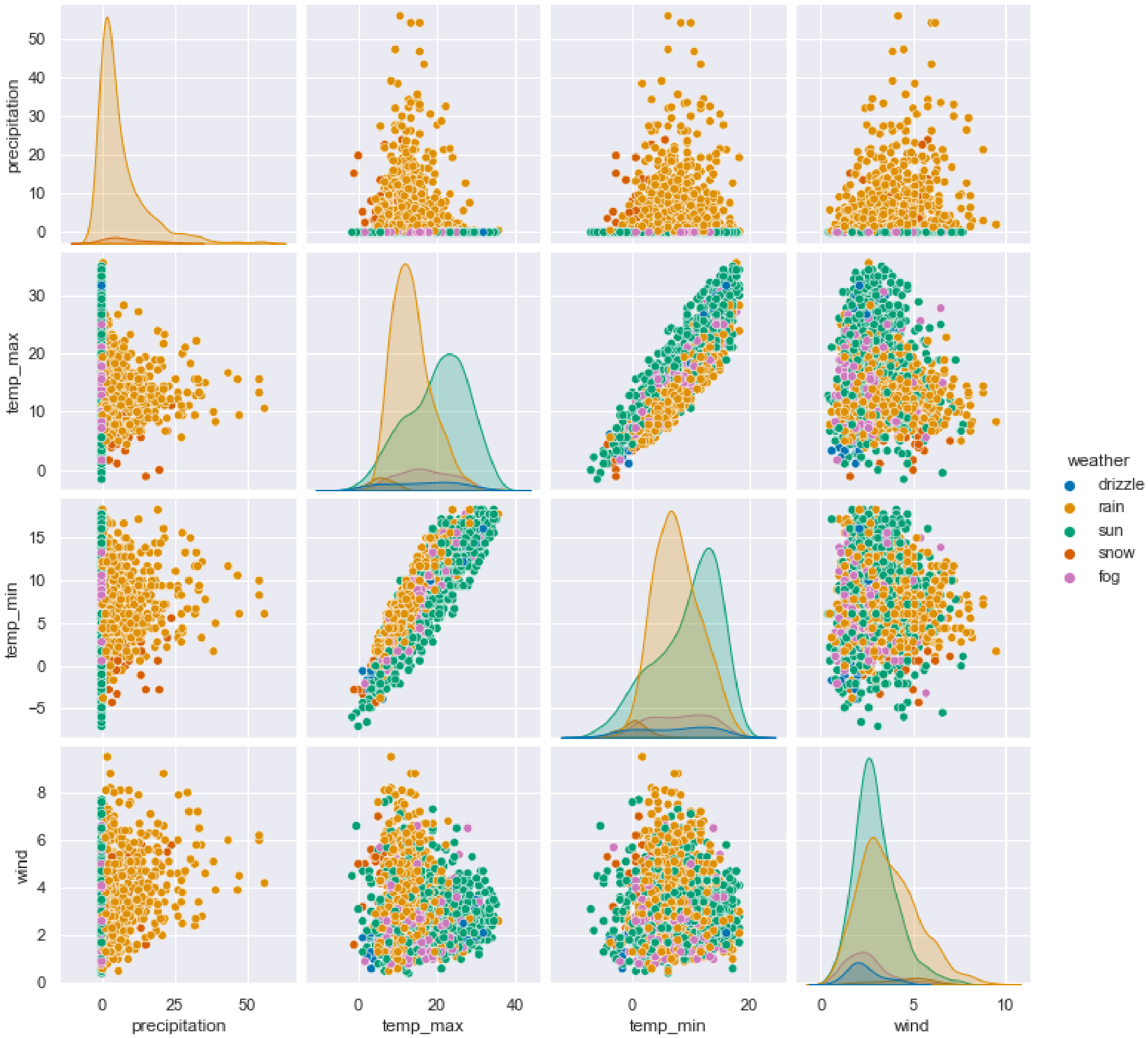
In [14]:
```
#Wind changing rate with respect to date
plt.figure(figsize=(18,8))
sns.set_theme()
sns.lineplot(x = 'date',y='wind',data=data)
plt.xlabel("Date",fontweight='bold',size=13)
plt.ylabel("wind",fontweight='bold',size=13)
plt.show()
```



Here we can see form the line plot that wind is almost stable in every month of the year. But arount the 7th month of each year, wind rate is little bit lower with compare to other months

In [15]:
```
# Pair plot for the data DataFrame
plt.figure(figsize=(14,8))
sns.pairplot(data.drop('date',axis=1),hue='weather',palette="colorblind")
plt.show()
```

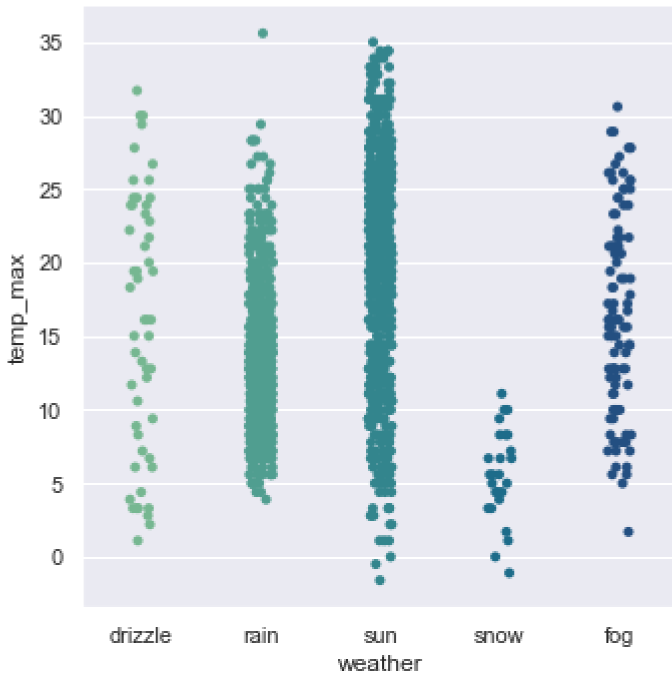
```
<Figure size 1008x576 with 0 Axes>
```



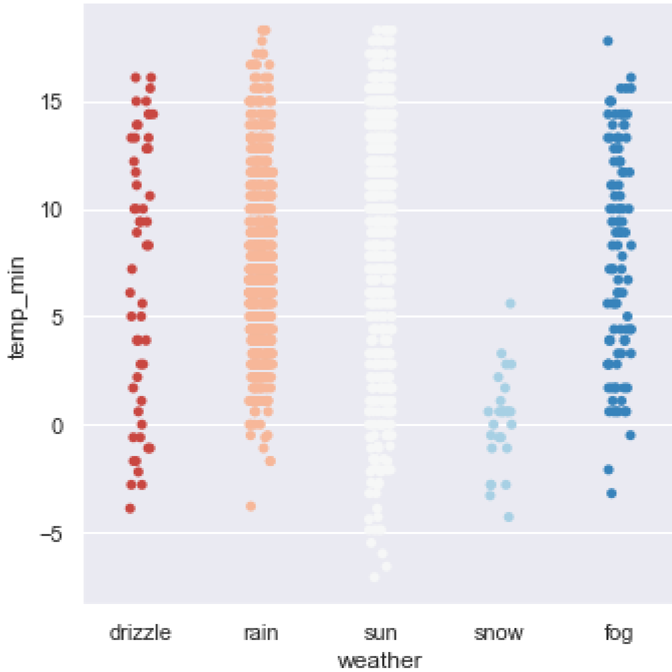
From the pair plot we find that, no such features can be separated from other species in all the features. But for precipitation, there is no other features except rain and snow. But, if we can ignore the precipitation we might not get better prediction. As most of the continious rainy days are rainy and sometimes snows are fallen. So we can not drop the precipitation features.

In [16]:
```
#Catplot: weather vs temp_max
plt.figure(figsize=(10,5))
sns.catplot(x='weather',y ='temp_max',data=data,palette="crest")
plt.show()
```

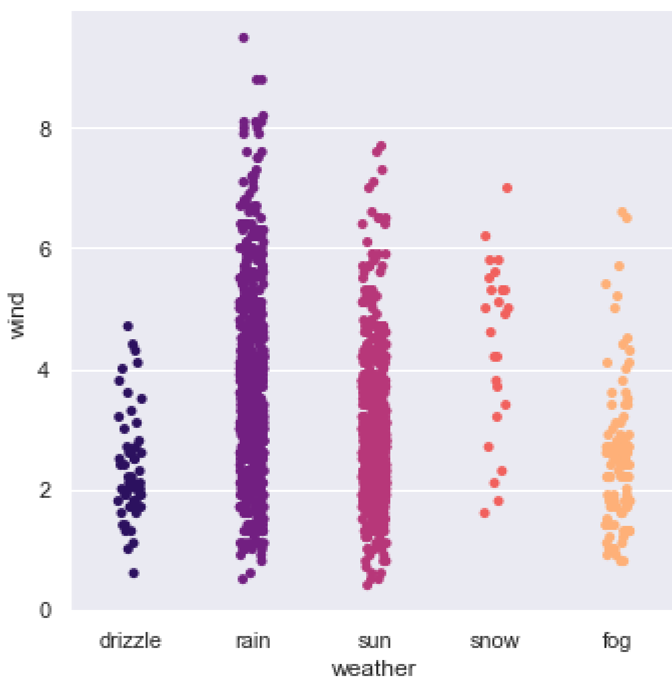
```
<Figure size 720x360 with 0 Axes>
```



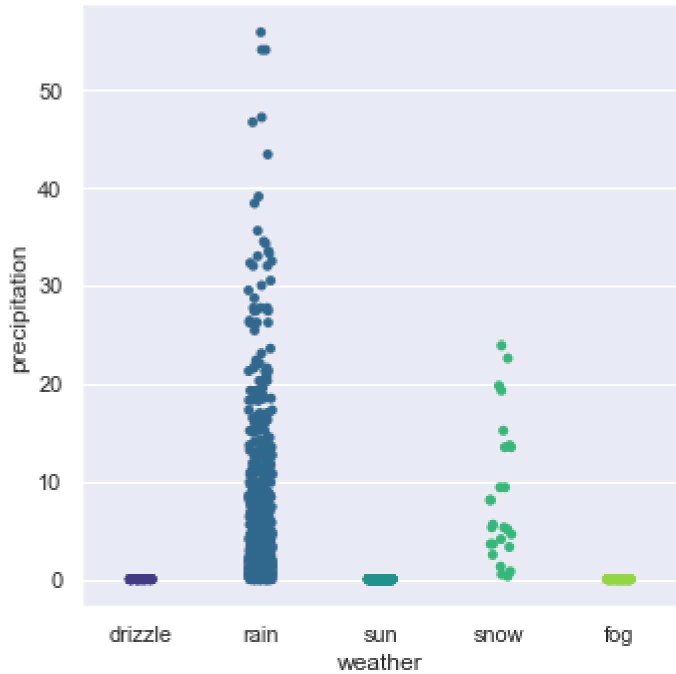
In [17]:
```
# Catplot: weather vs temp_min
sns.catplot(x='weather',y ='temp_min',data=data,palette = "RdBu")
plt.show()
```



In [18]:
```
#Catplot: weather vs wind
sns.catplot(x='weather',y ='wind',data=data,palette = "magma")
plt.show()
```

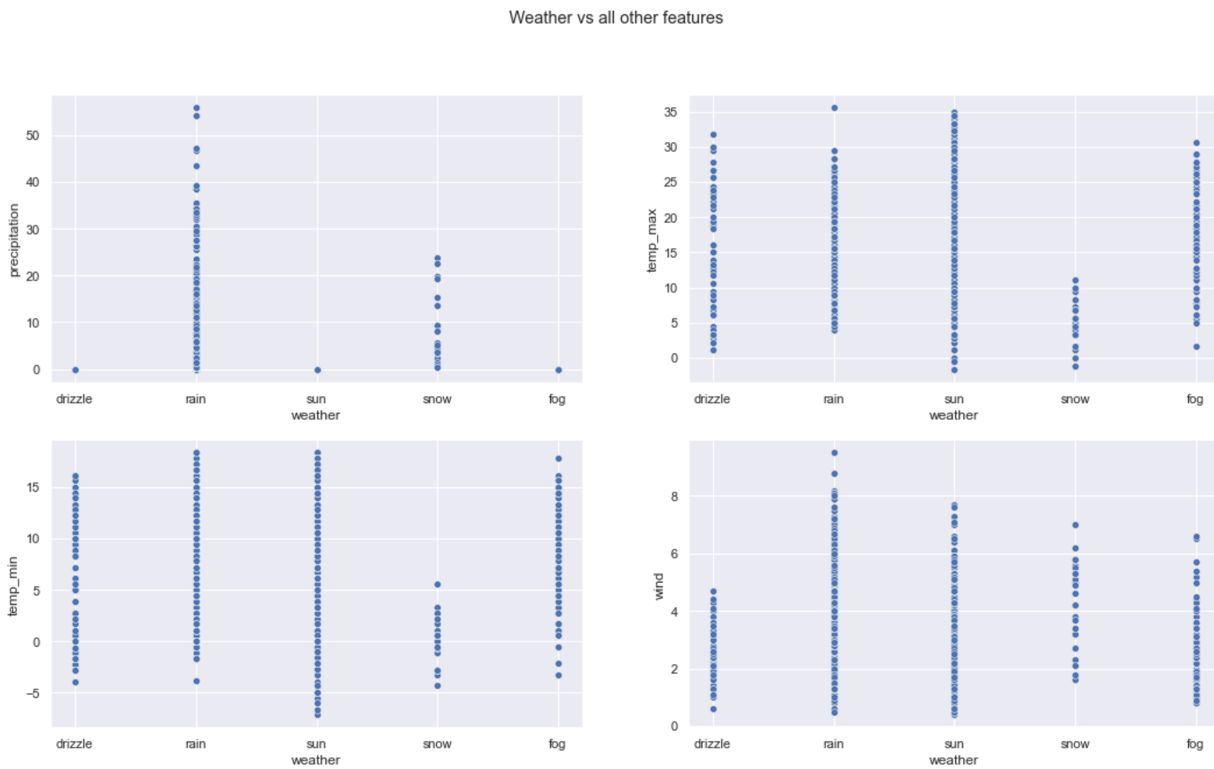


In [19]:
```
#Catplot: weather vs precipitation
sns.catplot(x='weather',y ='precipitation',data=data,palette = "viridis")
plt.show()
```



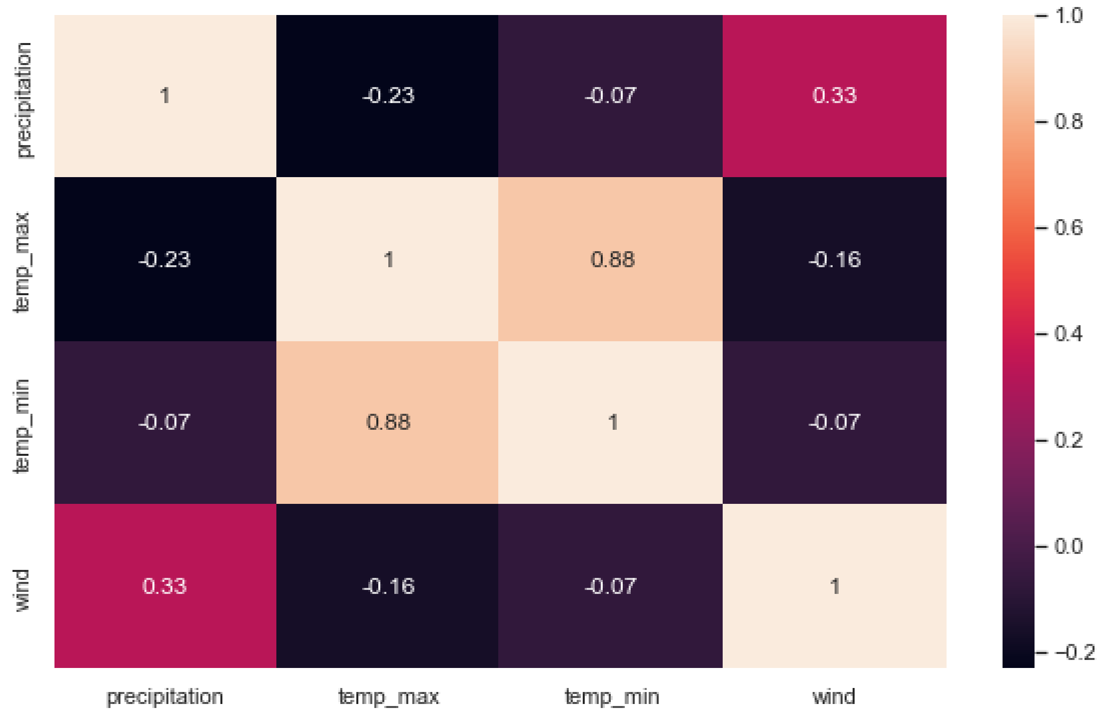
From the catplot we can identify that how much weather change for a perticular feature of the dataset

In [20]:
```
# Scatter diagram for each features
fig, axes = plt.subplots(2, 2, figsize=(18, 10))

fig.suptitle('Weather vs all other features')

sns.scatterplot(ax=axes[0, 0], data=data, x='weather', y='precipitation')
sns.scatterplot(ax=axes[0, 1], data=data, x='weather', y='temp_max')
sns.scatterplot(ax=axes[1, 0], data=data, x='weather', y='temp_min')
sns.scatterplot(ax=axes[1, 1], data=data, x='weather', y='wind')
plt.show()
```



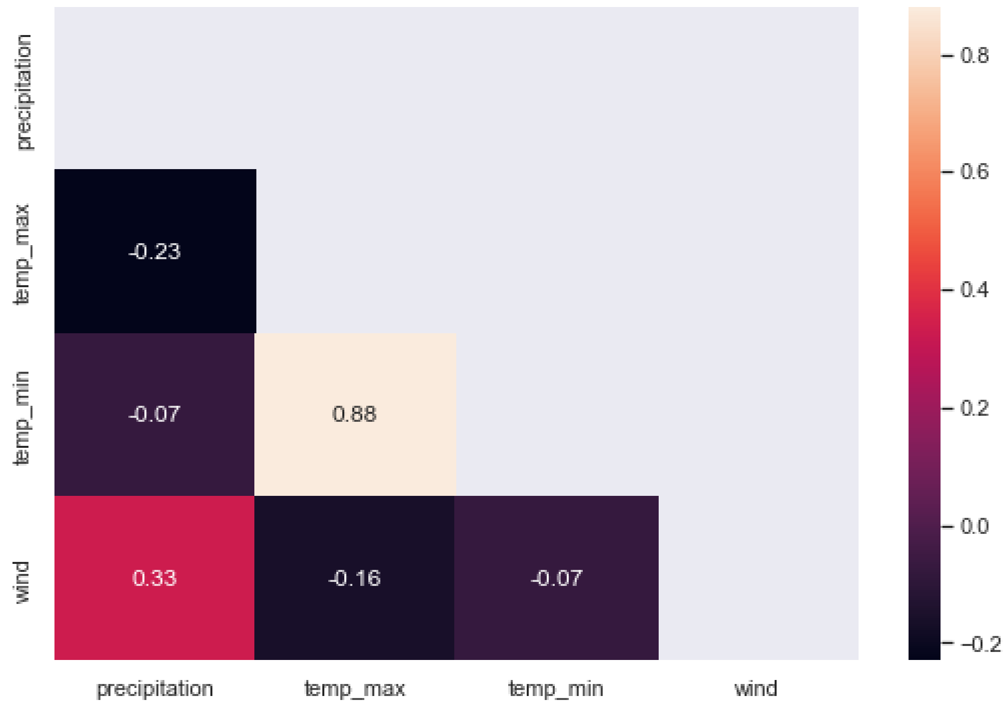
From the scatter plot we can identify that how much weather change for a perticular feature of the dataset

In [21]:
```
# Calculate the correlation between variables
correlation_matrix = data.corr().round(2)
plt.figure(figsize = (10, 6))
sns.heatmap(data=correlation_matrix, annot=True);# "annot = True" to print the va
```



In [22]:
```
# Steps to remove redundant values indices for the upper-triangle of array
mask = np.zeros_like(correlation_matrix)
mask[np.triu_indices_from(mask)] = True
plt.figure(figsize = (9, 6))
sns.heatmap(data=correlation_matrix, annot=True, mask=mask)
```

Out[22]: <AxesSubplot:>



As we can see, all the variables are stands strongly to predicts the accuracy of the weather.

In [23]:
```
data = data.drop('date',axis=1)
```

We don't need date to train our data as it is always increasing by one and also it is not a numerical value which we need to calculate accuracy and train our data

Create Features Matrix & Target Variable

In [24]:
```
#feature selection from dataset
x = data.drop('weather',axis=1)
x
```

Out[24]:

	precipitation	temp_max	temp_min	wind
0	0.0	12.8	5.0	4.7
1	10.9	10.6	2.8	4.5
2	0.8	11.7	7.2	2.3
3	20.3	12.2	5.6	4.7
4	1.3	8.9	2.8	6.1
...	...	...	...	...
1456	8.6	4.4	1.7	2.9
1457	1.5	5.0	1.7	1.3
1458	0.0	7.2	0.6	2.6
1459	0.0	5.6	-1.0	3.4
1460	0.0	5.6	-2.1	3.5

1461 rows × 4 columns

In [25]:
```
# target selection from dataset
y = data['weather']
y
```

Out[25]:
```
0       drizzle
1          rain
2          rain
3          rain
4          rain
         ...
1456       rain
1457       rain
1458        fog
1459        sun
1460        sun
Name: weather, Length: 1461, dtype: object
```

Split the dataset

In [26]:
```
#split the dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, rando
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1095, 4)
(366, 4)
(1095,)
(366,)
```

We have split our data set in 3:1 ration where 1/3 are stored for test. Rest of them will use for traning

Create Model

Support Vector Machine (SVM)

In [27]:
```
#Training and calculating accuracy for SVM
from sklearn import svm
from sklearn import metrics
model_svm = svm.SVC()
model_svm.fit(X_train, y_train)
y_prediction_svm = model_svm.predict(X_test)
score_svm = metrics.accuracy_score(y_prediction_svm, y_test).round(4)
print("----------------------------------")
print('The accuracy of the SVM is: {}'.format(score_svm))
print("----------------------------------")
score = []
score.append(score_svm)
```

```
----------------------------------
The accuracy of the SVM is: 0.7678
----------------------------------
```

Decision Tree

In [28]:
```
#Training and calculating accuracy for SVM
from sklearn.tree import DecisionTreeClassifier
model_dt = DecisionTreeClassifier(random_state=4)
model_dt.fit(X_train, y_train)
y_prediction_dt = model_dt.predict(X_test)
score_dt = metrics.accuracy_score(y_prediction_dt, y_test).round(4)
print("---------------------------------")
print('The accuracy of the DT is: {}'.format(score_dt))
print("---------------------------------")
score.append(score_dt)
```

```
---------------------------------
The accuracy of the DT is: 0.7404
---------------------------------
```

K Nearest Neighbours (KNN)

In [29]:
```
from sklearn.neighbors import KNeighborsClassifier
model_knn = KNeighborsClassifier(n_neighbors=3)
model_knn.fit(X_train, y_train)
y_prediction_knn = model_knn.predict(X_test)
score_knn = metrics.accuracy_score(y_prediction_knn, y_test).round(4)
print("----------------------------------")
print('The accuracy of the KNN is: {}'.format(score_knn))
print("----------------------------------")
score.append(score_knn)
```

```
----------------------------------
The accuracy of the KNN is: 0.6995
----------------------------------
```

Logistic Regression (LR)

In [30]:
```
from sklearn.linear_model import LogisticRegression
model_lr = LogisticRegression(solver='lbfgs', max_iter=500)
model_lr.fit(X_train, y_train)
y_prediction_lr = model_lr.predict(X_test)
score_lr = metrics.accuracy_score(y_prediction_lr, y_test).round(4)
print("---------------------------------")
print('The accuracy of the LR is: {}'.format(score_lr))
print("---------------------------------")
score.append(score_lr)
```

```
---------------------------------
The accuracy of the LR is: 0.8388
---------------------------------
```

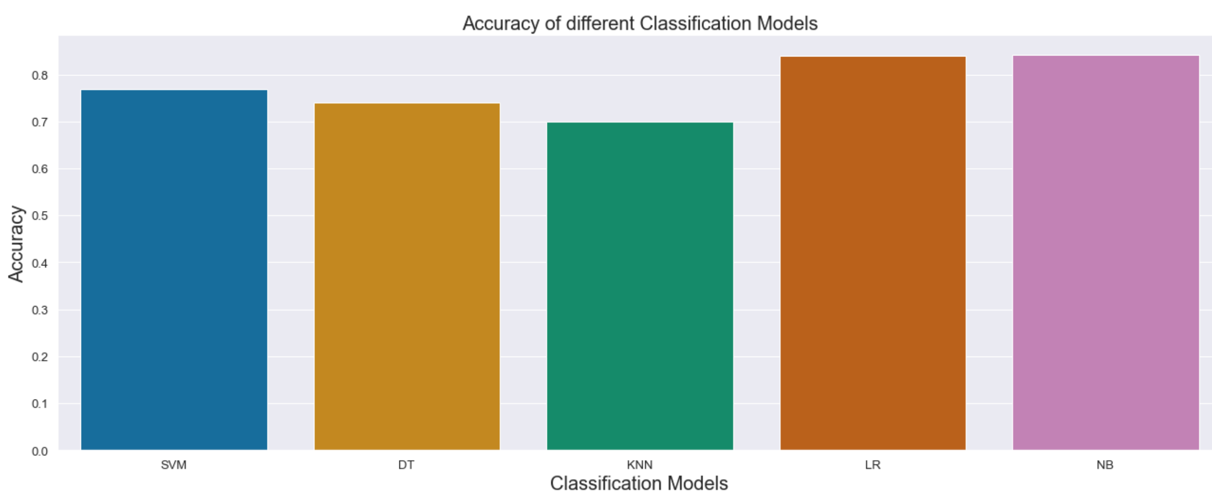
Gaussian Naive Bayes (NB)

In [31]:
```
from sklearn.naive_bayes import GaussianNB
model_nb = GaussianNB()
model_nb.fit(X_train, y_train)
y_prediction_nb = model_nb.predict(X_test)
score_nb = metrics.accuracy_score(y_prediction_nb, y_test).round(4)
print("---------------------------------")
print('The accuracy of the NB is: {}'.format(score_nb))
print("---------------------------------")
score.append(score_nb)
```

```
---------------------------------
The accuracy of the NB is: 0.8415
---------------------------------
```

Compare Accuracy Score of Different Models

In [32]:
```
#comparing traning model's accuracy
sns.set_style("darkgrid")
plt.figure(figsize=(22,8))
classifier = ['SVM','DT','KNN','LR','NB']
ax = sns.barplot(x=classifier, y=score, palette = "colorblind")
plt.xlabel("Classification Models", fontsize = 20 )
plt.ylabel("Accuracy", fontsize = 20)
plt.title("Accuracy of different Classification Models", fontsize = 20)
plt.xticks(fontsize = 13, horizontalalignment = 'center')
plt.yticks(fontsize = 13)
plt.show()
```



Discussion and conclusion

In this report, we did an analysis on a dataset known as the 'weather prediction'. Here, we developed 5 different of classifier model which are Support Vector Machine, Decision Tree, K Nearest Neighbours, Logistic Regression and Gaussian Naive Bayes. We can see that the lowest accuracy model is K Nearest Neighbours (KNN) which accuracy is 0.6995. The Decision Tree (DT) classifier accuracy is 0.7404 which is better then KNN. Support Vector Machine (SVM) acuracy is 0.7678, Logistic Regression (LR) accuracy is 0.8388 and Gaussian Naive Bayes (NB) accuracy is

0.8415. So, the highest accuracy rate is 0.8415 which is Gaussian Naive Bayes (NB) classifier model. As a result, we can say that the Gaussian Naive Bayes classifier is the best use for this dataset model. The Gaussian Naive Bayes classifier model's accuracy is below 90% because of the dataset. It might perform better if we can train these model on a larger dataset.

So in our opinion, depending on this dataset, the Gaussian Naive Bayes classifier is best for predicting the weather. Although, for a larger dataset other model may perform better.