What EDA did you do on the dataset?

First, I just did df.describe() to get the basic statistical summary of the dataset.

```
df_train.describe()
```

[3]:

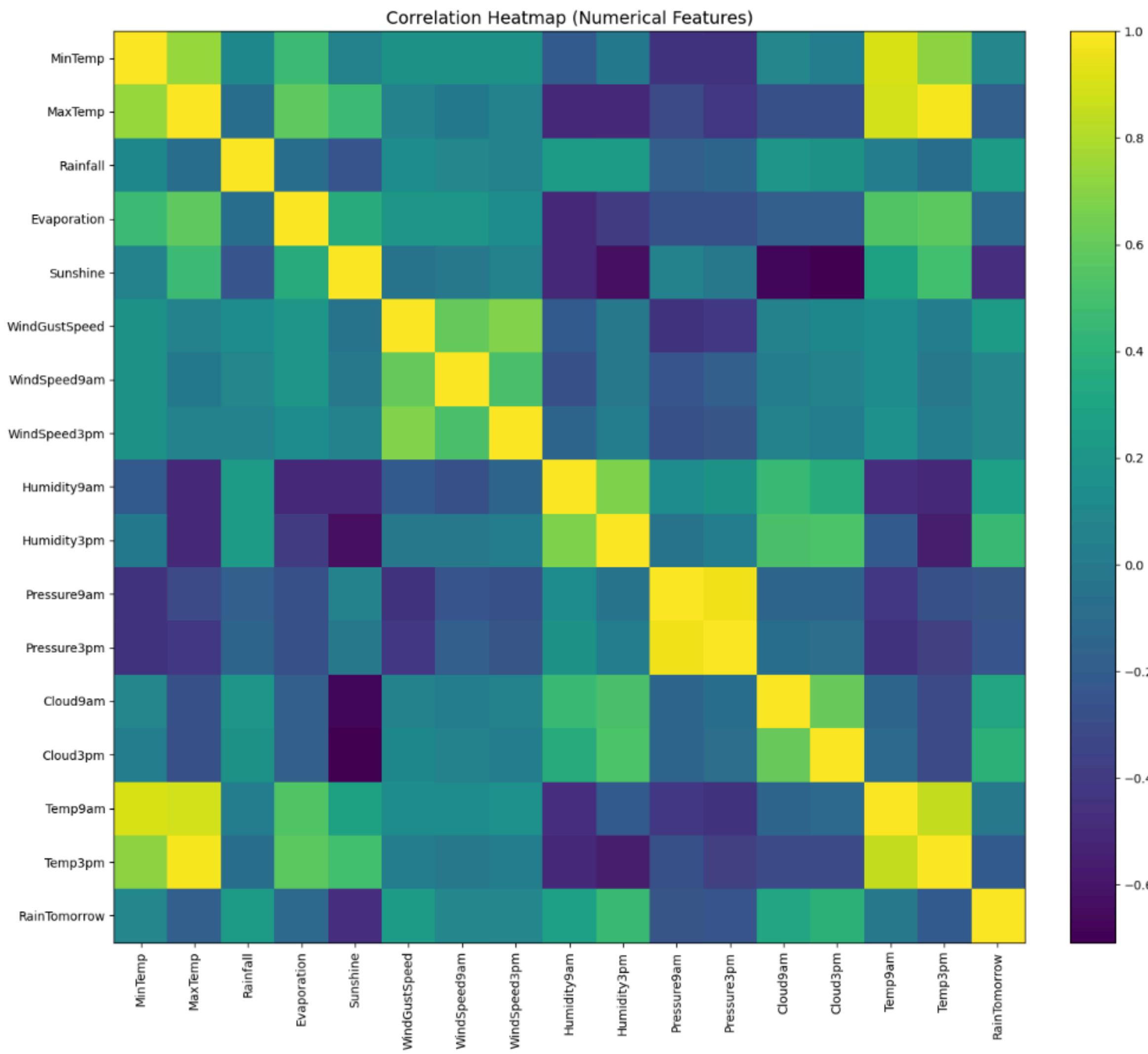
	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	
count	115853.000000	116112.000000	115173.000000	66498.000000	60800.000000	108746.000000	115258.000000	114193.000000	114903.000000	113391.000000	104858.000000	1
mean	12.217997	23.159362	2.488305	5.450544	7.518002	40.160383	14.033325	18.689140	69.153477	52.072572	1017.539227	
std	6.408990	7.120924	8.793658	4.175225	3.813882	13.697858	8.894625	8.846438	19.050740	20.984941	7.135135	
min	-8.500000	-4.800000	0.000000	0.000000	0.000000	6.000000	0.000000	0.000000	0.000000	0.000000	980.500000	
25%	7.600000	17.900000	0.000000	2.600000	4.700000	31.000000	7.000000	13.000000	57.000000	37.000000	1012.800000	
50%	12.000000	22.600000	0.000000	4.600000	8.300000	39.000000	13.000000	19.000000	70.000000	52.000000	1017.500000	
75%	16.900000	28.200000	0.800000	7.400000	10.600000	48.000000	19.000000	24.000000	83.000000	66.000000	1022.300000	
max	31.400000	48.100000	371.000000	145.000000	14.300000	135.000000	130.000000	87.000000	100.000000	100.000000	1041.000000	

None of the features had an abnormally high standard deviation or looked out of place, so I ruled out the process of eliminating outliers.

In the discussion session about this hackathon we were told multiple times to look for data leakage, and so I tried to find any one feature that would sway the prediction of the model. I did this by (on chatgpt's suggestion) training the model with only one column for every column. I then saw the accuracies that gave, and no one column's accuracy exceeded around 80%. I tried a few more of chatgpt's suggestions to find data leakage and none of them gave any substantial result.
I even fed the entire dataset to perplexity and asked it to tell me its analysis and whether there is any leakage I should be worried about, and it told me the data was fine apart from having a lot of missing values.
Eventually I ran out of time and couldn't find the leakage we were warned about.
(I want to include a screenshot of this but I deleted the cells in which I did this in before submitting the notebook)

Correlation heatmap: (done after the deadline)

From here, we can see that pressure 9 am has a VERY strong correlation with pressure 3 pm.
Maxtemp and temp 3 pm are also very highly correlated.
These highly correlated features might've messed with the training of the dataset.
If I'd actually done this heatmap in time, I would've removed either pressure 9 am or pressure 3 pm and probably temp 3 pm. (even temp 3 pm and 9 am are highly correlated.)
Sunshine is very negatively correlated with humidity.
Both cloud 9 am and cloud 3 pm have a highly negative correlation with sunshine.



Preprocessing Techniques

- First, I deleted all null values because I thought we had enough data - next I replaced the numeric nulls with medians - eventually realized because I was using lightgbm it inherently handled nan/nulls so I didn't touch the nulls and that boosted my score.
- One-hot encoded all the non-numeric columns.
- Grouped "Date" into seasons instead as I felt that would affect the rain more - this boosted my score from 92 ish to my final score of 97 ish.

Which model did you use and why?
Also explain what tweaks you made on the model over your submissions.

I was settled on using gradient boost because I read that tree based models work well for large tabular datasets with many features and boosting was a very efficient and accurate way to implement them, which each tree correcting the previous.

So, the first approach I tried was using the XGBoost model. Worked decent with an 85% accuracy but had a bad precision and recall, which means it wasn't predicting the rainy days well due to imbalance in training data.

I tried to get rid of this imbalance by introducing synthetic data using smote but that did more harm than good.

Tried using scale_pos_weight to combat imbalance in training data and that actually helped.

Tried CatBoost instead of XGBoost (with weights). It was pretty easy to implement since no one-hot encoding was required; it inherently handled categorical features. However, it was painfully slow, and each run took 10+ minutes, and the score wasn't good enough to make up for it.

Finally, I tried weighted LightGBM, which worked best of all. I didn't even need to operate on the nulls because the tree automatically took care of that.
(The parameters in my lightgbm model were chosen by chatgpt with specific emphasis on avoiding overfitting)

Tried Neural Networks, but it kept giving an error (I think this was due to my mishandling of NaNs); eventually I ran out of time and couldn't make a working submission.

Explain your view on why your model failed or performed better than expected on the private leaderboard test data.

Well, my final model surpassed my expectations on the private leaderboard, and I think this was owed to the heavy emphasis on avoiding overfitting in the parameters I fed to the model.