Group 2 Challenge 1

Following are sections illustrating our work predicting weather

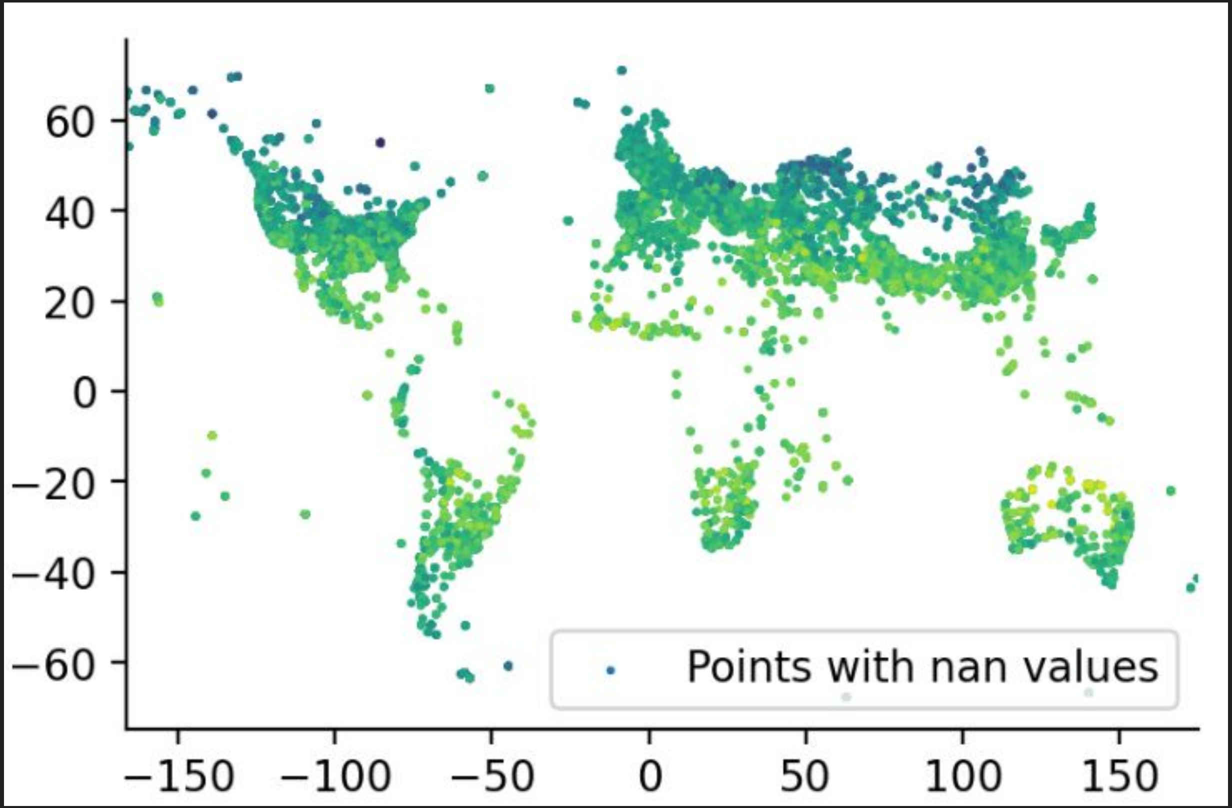
Data Analysis Objectives

- Pinpoint possible missing data
    - Find out the importance of NaN-values
    - Check for covariate shift
- As there are 111 features in total, find out if we can drop any
    - E.g. look for high correlation and usefulness
- Choose candidate columns for feature engineering

Data Analysis - NaN-values

- We suspected that the NaN-values were not distributed equally around the globe, eg. that we would loose specific information by dropping them
- By plotting the distribution of NaN-values we concluded that it was safe to drop all NaN-rows, as the plot showed us that the NaN-values had a similar distribution to the non-NaN-values. This indicates that the NaN-values are randomly distributed in the training set
- In addition since our training dataset was that large (~2M), and number of nan-rows that small (~130k), dropping nans would not lead to shortage of data.
- The picture on the next slide shows the distribution of NaN–rows with respect to latitude and longitude

Data Analysis - NaN-values-distribution



Data Analysis - Correlated Columns

- We wanted to identify which columns that contributed to the same information for the temperature, i.e. which columns were highly correlated
- Worth noting is that we’re only checking for linear correlation, which is why we might get a better RMSE when including all the columns
- However, not including all the columns will reduce the training time
- Started by finding the correlation-matrix between every column
    - I.e; a table with a value to each pair of columns that described how well the two columns were correlated
- We could then extract potentially redundant columns

Data Analysis - Covariate Shift

- From the plots given in the baseline code, it's evident that we face the problem of covariate shift, i.e. that test data distribution differs from train data distribution
    - In this case the problem is that the training-set has few equatorial data-points
- To begin with, we therefore did an analysis on how much effect this covariate shift had on our score. We generated one training set that excluded all equatorial data points (points between -10 and 10 latitude), and validated it on all the equatorial data points in the training set. We then generated a new training and validation set of the same size but this time having a random split

Data Analysis - Covariate Shift

- Local RMSE Score on unseen (equatorial) data: 1.7989
- Local RMSE Score on seen (random split) data: 1.622
- This was not as critical as we first thought, and we therefore focused more on other parts of the assignment. However, it can be a greater problem given a different test set and is hence something that should be solved in the future

Data Preprocessing Objectives

- Extract the necessary columns
  - I.e. non-redundant and most important to ‘fact-temperature’
- Feature engineering

Data Preprocessing - Dropping Correlated Columns

- From the correlation matrix we constructed in the data analysis part, we could retrieve a list of a given column’s most correlated columns
- Hence, we extracted the pair of columns that had an absolute correlation value of more than or equal to 0.95 (highly correlated), and made a list for each column (parent) with its “highly correlated” columns (children)
- Next, we dropped the children columns for the parent column with the highest number of children
- Our best result was still using all the columns, and not dropping the correlated ones

Data Preprocessing - Covariate Shift

- One possible solution for the lack of equatorial data points, hereafter referred to as *minority*, in the training set is upsampling
    - Upsampling is a technique to generate synthetic data based on the few points we already have
- An easy way was to copy and multiply the rows in the training set that where between -10 and 10 latitude (inside equator)
- We tried doing this by multiplying the rows by 5. However, this did not yield better results
- There are however other techniques for upsampling that could be helpful (like SMOGN)

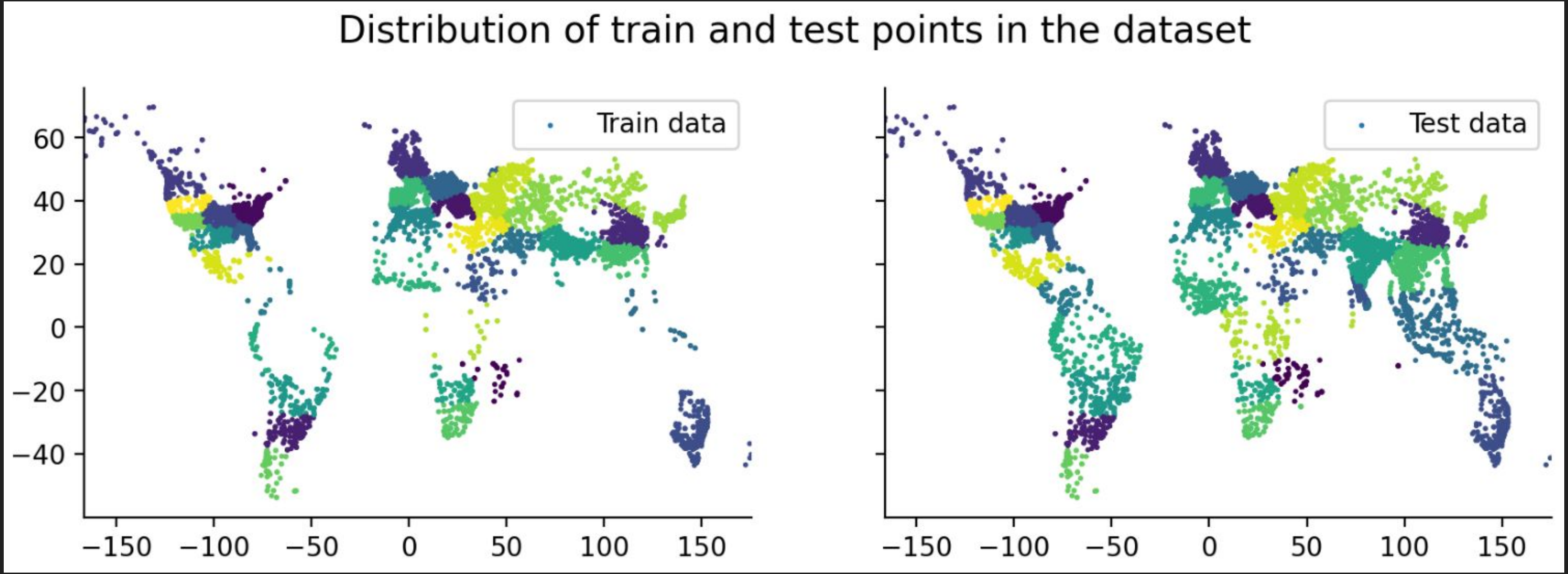
Data Preprocessing - Feature Engineering on Time

- By feature engineering ‘fact_time’ to retrieve separate columns for ‘Hour’, ‘Day’ and ‘Month’ the training process will be able to find more accurate patterns in regards to ‘fact_temperature’
  - This is because a unix timestamp itself is continuous and the model will not learn anything off a column where every row has a unique value.
  - We also discussed a potential problem about time zones, e.g. that hour 12 in France is different from hour 12 in Australia, and that winter in Australia is when we have summer in France.
    We did not have enough time to investigate this, and concluded that the model managed to represent it in a way by combining hour and longitude in the tree-structure.

Data Preprocessing - Feature Engineering on Clusters

- We created a new feature representing the cluster-number
- Clusters are created with sklearn’s KMeans model. The model is created using the training set’s latitude and longitude with ‘fact_temperature’ as target. The KMeans model is then used to predict the cluster-value of every point in both train set and test set
- To make the KMeans model prioritize clustering on latitude over longitude we scaled this column, which gave more appropriate clusters
- The clustering-feature gave a substantial improvement in model performance, and was listed as XGBoosts’ top features
- The clusters groups the datapoints together, where each cluster has about the same temperature. You can see the clusters, i.e. different groups of temperatures, on the next slides

Data Preprocessing - Feature Engineering on Clusters



Data Preprocessing - Feature Engineering on Lat-long

- The XGBoost model allows us to rank the columns based on their contribution to the training. From this function, we have gathered that latitude and longitude have the most impact during training
- We did try to “gridify” latitude and longitude by grouping multiple latitudes into wider “latitude grids” and likewise for longitude, however it deemed not to be lucrative as we did not get a better RMSE.
- A further feature engineering on these columns might therefore be beneficial

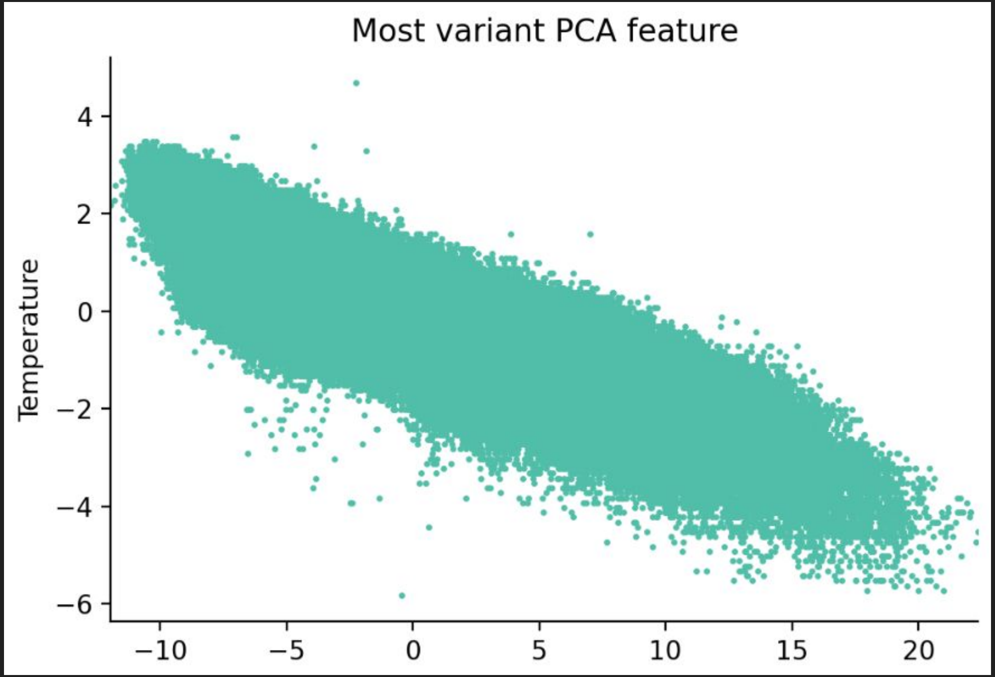
Data Preprocessing - Principal Component Analysis

- Since our dataset had a lot of features it was only natural to explore methods for dimesionality reduction
- PCA transforms the features in the dataset to the features that capture the largest amount of linear variance in the dataset
- We transformed our training set to capture 90%, 95%, and 99% of the variance, and that left us with 33, 43, and 61 features respectively

Model:XGBoost	Test RMSE	Training and inference time (s)
90%	0.249	660.64
95%	0.247	872.51
99%	0.242	1178.16

Data Preprocessing - Principal Component Analysis

- We graphed the feature that explained the most variance in the training dataset against our target variable
- Explained variance: 22.9%



Modelling Objectives

- Choose fitting model
- Optimize parameters

Model and Model Tuning

- We decided from the start to use XGBoost
  - Easy to use and really good performance
  - The possibility to train with GPU also made it fast
  - Could throw nan-values at it without any problem as they are automatically handled
  - We should be aware though, that XGBoost is not ordinarily deterministic which can yield different results for different iterations
- We used Optuna in combination with KFold for hyper-parameter-tuning
  - Optuna is a framework that lets you test out different hyper-parameter-values in combination
  - We used KFold with 4 folds in every Optuna-trial in order to not overfit.

Model and Model Tuning

- Other modelling approaches were also tested:
  - We implemented an ensemble (a collection of multiple models) of LGBM, Catboost and XGBoost, with sklearn’s RidgeCV as meta-model (the meta model is a “simple” model that tries to learn from the predictions by the model-collection).
  - This has given us good performance in previous projects, but for this project only using XGBoost provided the best results. (Note that most of this code was provided by our own previous projects)
  - In the notebook this code is commented out and can be found under “Test of ensemble model”