The notebook starts by introducing the c*urated_data_1month_2010-2022_nonans.csv* dataset that will be used. It explains each field and gives the distribution stats for each field

A particular piece of analysis uses histograms to look at the skew that interprets the data as having a left skew. The follow up says that for frost and snow the lowest levels are overrepresented.

This step is called Exploratory Data Analysis (EDA).

The notebook then plays around with reading in a dataset, using numpy to put it in tabular array form and selecting feature/label subsets

Splitting the data is taken a step further into test, train and validation. Again using seeded random splits.

The validation test is used to test the performance of the models using different hyperparameters. This allows for the hyperparameters to be optimized before touching the unseen test dataset.

The notebook then explores training an SVM model whilst using grid search method to optimize the hyperparameters on the validation set

Validation searching on the validation set is done before the training set is utilised to train the final model.

The next section works on scaling inputs. The professor has explained before that this is step that needs to be executed in the assessment

A module called StandardScaler from sklearn is used in the notebook and the documentation listed for reading

Standardizing has a number of benefits.

It is good for convergence as large values can create large jumps in a gradient descent capacity. This can lead to slower convergence and instability.

Large values naturally will have more impact on the network and result in dominating features rather than learning the true relationships.

Some activation functions, such as sigmoid and then, have an “active region”. Extreme values fall outside of this range and can result in vanishing or exploding gradients.

In tandem with regularization, if features are not scaled then regularization may unfairly affect weights with associated larger features, even if their importance is not greater. Hence, scaling also helps with overfitting and generalization.

Important, best practice is to fit a scaler to the training data and then apply that scaled fit to the test/validation set. Applying the fit to the whole dataset will result in “data leakage” coming from the test data which is meant to be unseen throughout the training process.

Standard scaling means each feature now has a mean of 0 and standard deviation of 1.

There are a number of different scaling methods to choose from.

Scaling will always affect the model outcome so inputs should always be scaled. This is important for the assessment.

Scaling is a preprocessing step so it applied before any modelling takes place.

Scaling can change the loss function performance and the optimal hyperparameters.

Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance).

For instance many elements used in the objective function of a learning algorithm (such as the RBF kernel of Support Vector Machines or the L1 and L2 regularizers of linear models) assume that all features are centered around 0 and have variance in the same order. If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected.

Next, the notebook is splitting the target value which is a continuous variable into 4 classes. This may be important for the assessment as the target variable will likely come as a continuous variable. Though the decision to split into or will be based on the model design.

The splitting code used numpy.place() as the operator. The chose splits were not explained but the data distribution should likely determine this.

The notebook then trained a Logistic Regression model

Next other evaluation metrics are explored, namely the F1 score and the confusion matrix

Need for evaluation models are the truth labels and the prediction classes

More details about evaluation metrics are covered in weeks 5/6

The F1 Score is a single metric that combines **precision** and **recall** into a balanced measure of a model's accuracy.

Specifically, it is the **harmonic mean** of precision and recall, meaning it penalizes models that have a low value in either precision or recall more heavily than a simple average would. It's particularly

useful for evaluating classification models, especially when dealing with **imbalanced datasets**, where a high accuracy alone can be misleading.

Recall (Sensitivity or True Positive Rate): Of all the **actual positive cases** in the data, how many did the model correctly identify? It measures the model's ability to find all relevant instances.

Precision (Positive Predictive Value): Of all the cases the model **predicted as positive**, how many were actually positive? It measures the accuracy of the positive predictions.

This means the F1 score is heavily concerned with the prediction of positive instances.

A **confusion matrix** is a table that summarizes the performance of a classification model by showing the counts of **correct and incorrect predictions** broken down by each class.

It specifically highlights four key outcomes:

- **True Positives (TP):** Correctly predicted positive cases.
- **True Negatives (TN):** Correctly predicted negative cases.
- **False Positives (FP):** Incorrectly predicted positive cases (Type I error, "false alarm").
- **False Negatives (FN):** Incorrectly predicted negative cases (Type II error, "missed opportunity").