RVAS Images Classification

Yesica Lopez
Asna Shafiq
Jonathan So
Vicram Uppal

Overview

- Background
- Summary of Project Goal
- Data Preparation
- Classification Models
  - K Nearest Neighbours
  - Support Vector Machine
  - Random Forest
- Model Comparison
- Conclusion & Lessons Learned

Background

- The Meteorological Service of Canada (MSC) of Environment Canada collects weather and climate data across Canada to provide authoritative information used to make health and safety decisions.
- To ensure the reliability of these sensor measurements, cameras were set up at remote unmanned stations to capture images of the sensors to detect anomalies in the data and for continued maintenance of the station
- When anomalies are detected in the data, technicians visually inspect the images to manually verify the proper operation of weather stations and ensure that the collected data is accurate
- However, as there are hundreds of images being generated by each station every day, this leads to increased workload for staff

What is an RVAS image?

- Remote Video Acquisition System
- Typical images are 1080x800 pixels
- Stations transmit between 1-4 images per hour
- The cameras' field of view varies greatly: capturing the full compound vs pointed at a specific sensor

VRA

VKC

VOC

What does an observation from the site looks like?

- Typical sensor will be reporting Snow On Ground (SOG) value, in addition to Temperature, Wind, Relative Humidity and Precipitation.

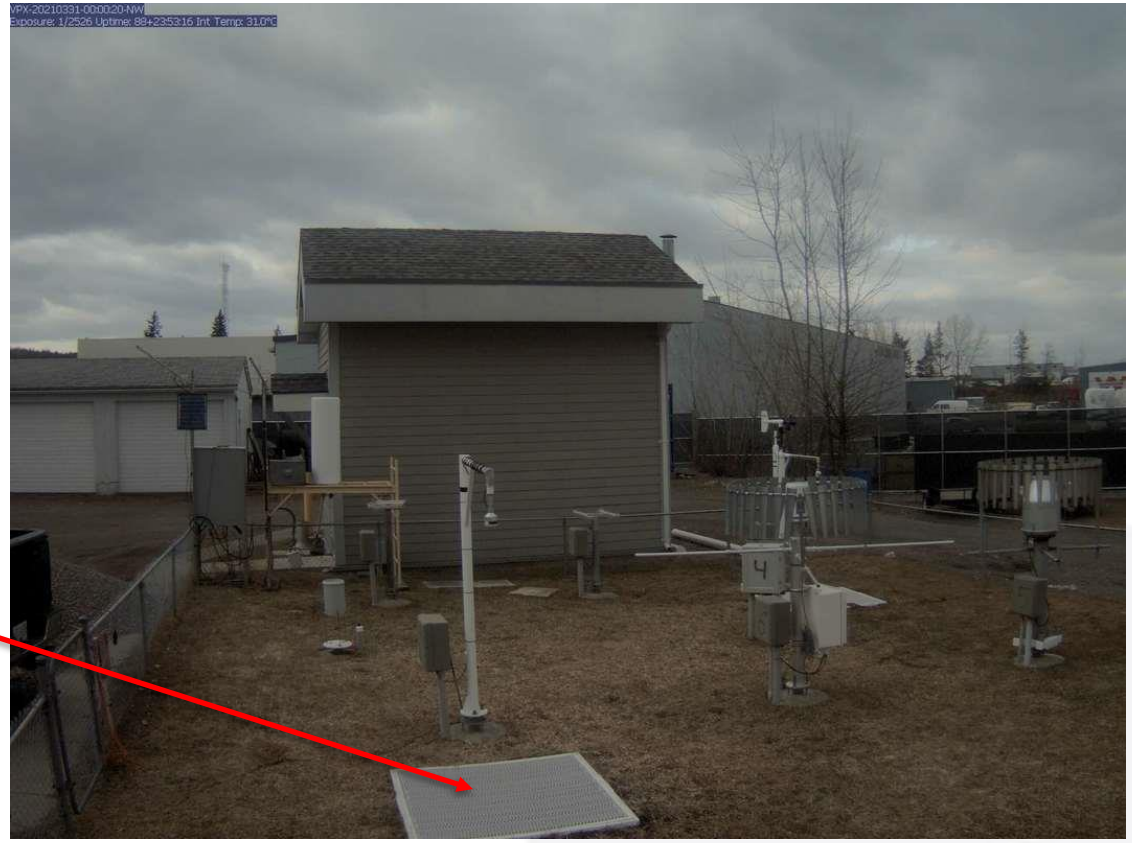
Obs Time (Z)	T1 (°C)	T2 (°C)	RH (%)	2m Wind Dir (deg)	2m Wind Spd (kts)	WG1 (mm)	WG Amt (mm)	TBRG Amt (mm)	SOG1 (cm)	Stn Prs (hPa)
Mar 31 1800	4.7	4.3	62	305	2	459.3	0	0	6	948.8
Mar 31 1700	3.4	3.1	66	348	1	459.3	0	0	7	949.6
Mar 31 1600	0.9	0.6	83	55	0	459.3	0	0	6	950.2
Mar 31 1500	-0.6	-0.8	91	310	0	459.3	0	0	6	950.6
Mar 31 1400	-1.0	-1.2	93	351	0	459.3	0	0	7	950.5
Mar 31 1300	-1.3	-1.5	93	39	0	459.3	0	0	7	951.0
Mar 31 1200	-1.5	-1.6	94	57	0	459.3	0	0	7	951.5
Mar 31 1100	-1.9	-2.0	94	216	0	459.3	0	0	7	951.6

Objective

- Build an automated image processing system that flags any abnormal site conditions
  - Abnormal condition is when a reported observation does not match the site condition captured by the camera
- For this project, we are focusing on Snow On Ground value, and classifying the images into "Snow" or "Clear"

Snow on ground sensor

- A typical automatic snow sensor measures the changing distance between the sensor and the platform
- This can lead to many potentially erroneous reporting. Such as animals sleeping on the platform or snow blown off the platform due to high wind



DATA PREPARATION

Data preparation for the RVAS images

- The dataset used are images from February 2021, taken at the top of the hour for 50 stations across Canada
- The images were resized from 1080x800 to 50x50
- The resized RGB images were transformed from a 3D array 50x50x3 into a 1D array of 7500 values between 0-255

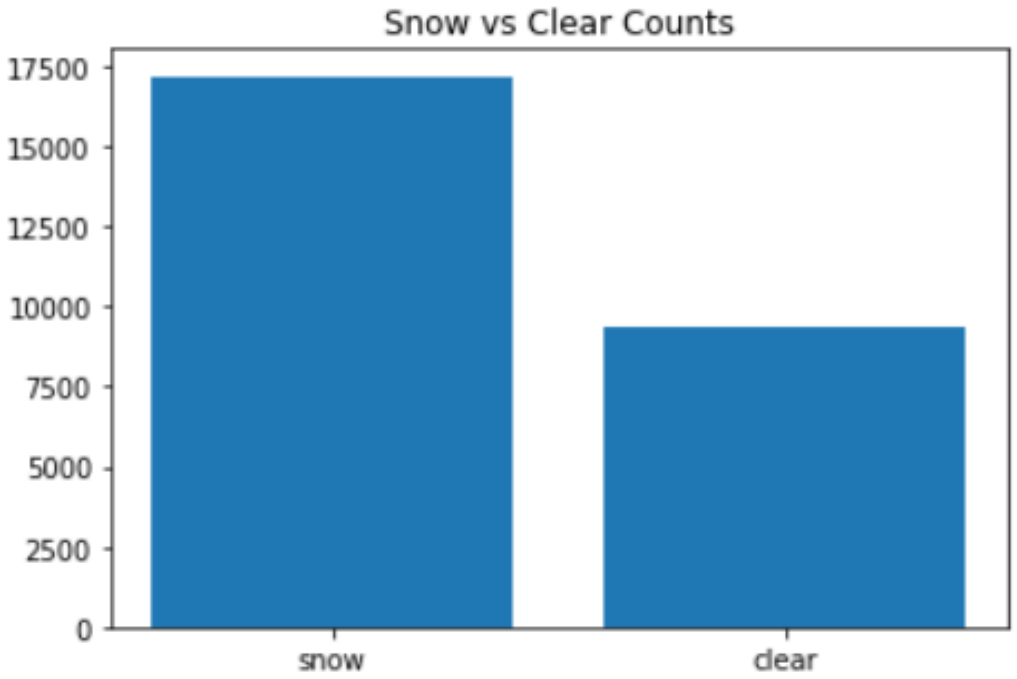
Data preparation (Labeling)

- The target label was generated by matching the weather observation from the time of the image and inspecting the Snow On Ground value
- With the limited time we only used the Snow on Ground value from the observations to label the images
- The target labels was assigned based on the following rules:
  - 'snow' if snow depth was > 1cm
  - 'clear' if snow depth was < 1cm
  - 'msng' if snow depth value was not available
- Due the potential inaccuracy of the measuring method this introduced inconsistencies to our model

Data preparation (Removing images)

- Images with 'msng' label were removed
- Images that are too dark for processing were removed from the dataset
  - The overall mean value of the pixels was used to identify images that are too dark for processing
  - Each pixel value ranges from 0-255. Black has a value of 0, so a low mean value implies a very dark image
  - Any images with a mean value <= 50 were removed
    - This threshold was picked based on sampling images at a 5-point interval until the images did not look completely black

Data Summary



Summary of our dataset

- Total images (38601)
- Images with MSNG labels (2986)
- Images that are too dark for processing (9053)
- Images used for training (26562)
- Images with 'Snow' 17117 (~65%)
- Images with 'Clear' 9385 (~35%)

Split train-test data

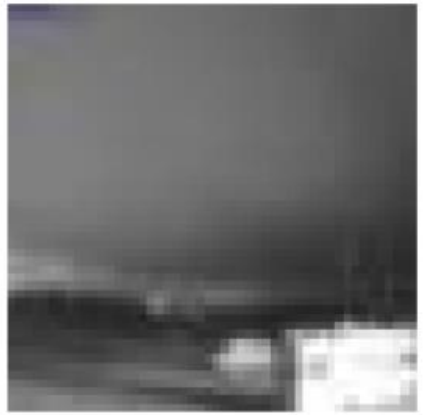
- The data was split 80-20 between train and test
- One station (WJR) was excluded from the training/test dataset so we could use it solely for testing knowing no images from that station were used for modelling
  - This would be representative of any future stations being predicted with the already built model

Sample Resized Images

snow

snow

clear



clear

snow

snow

Example of issues with the image dataset

- This image was labeled as 'snow' since the sensor reported a snow depth of 1.033554 cm
- Majority of the image does not have any snow
- The area that the snow sensor is measuring is outside of the image

Example of issues with the image dataset

- This image was labeled 'clear' as the sensor reported 0.917999 cm
- There is visible snow on the ground for most of the image
- The area that the snow sensor is measuring is outside of the image so we can't see how much snow is on the platform

Camera location in relation to the sensor

- The Camera positions varies for each station
  - Some stations have the camera pointing at the sensors
  - Others are pointing away from the sensors
  - The images may not always reflect the snow condition on the platform
  - Wind could have blown off the platform or there could be ice accretion on the platform which could also affect the automatic sensor

MODEL BUILDING

Development Environment

- The resizing and labelling was done locally via python scripts
- The final processed images were uploaded to a GitHub repo that Google Collab can pull from
- Model training and testing was done using the Google Collab platform

Aside – Storing Models

- Google Colab disconnects are not fun :(
- Being able to save/dump your trained model to a file to reload after in no time is the best
  - ```
    from joblib import dump, load

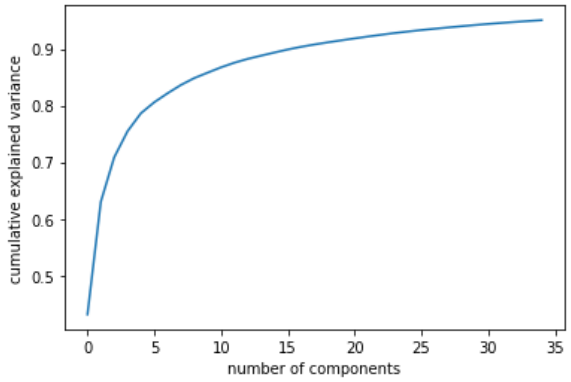
    dump(my_model, 'my_model.joblib')
    my_model = load('resources/trained_models/my_model.joblib')
    ```
    ```

Classification Models

- We evaluated 3 different supervised classification algorithms
  - K-Nearest Neighbours
  - Support Vector Machines (SVM)
  - Decision Trees – Random Forest

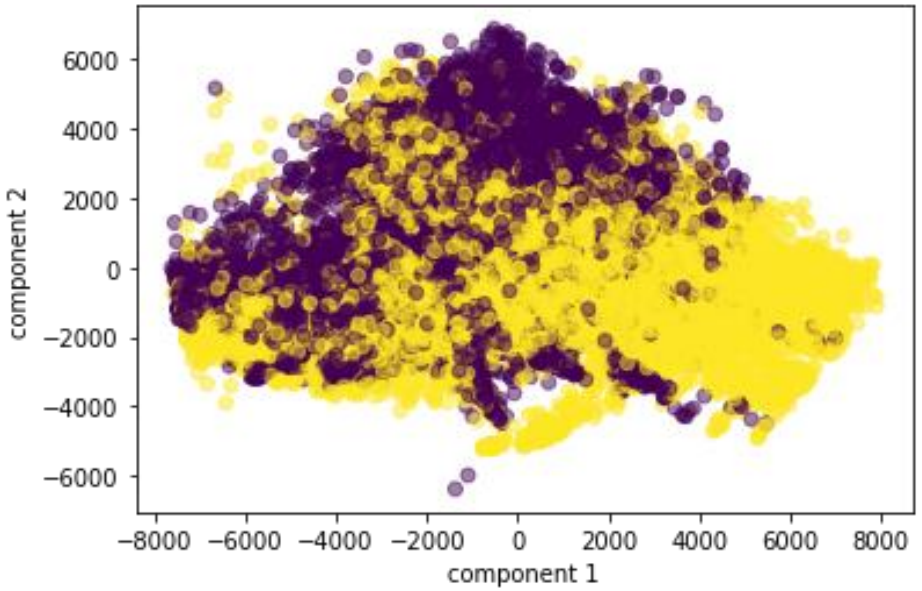
Dimensionality Reduction - PCA

- Dealing with RGB images 50x50 meant we had 7500 features (a lot!)
- Ran Principal Component Analysis (PCA) to find the components that would retain 95% of the dataset's variance
- This resulted in a reduction from 7500 -> 35 features!
  - `[0.43130397 0.19888944 0.07839172 … ]`



PCA – Visualization Aside

- Ran PCA with n_components=2 in order to try visualizing the data (which we couldn't do at the higher dimensions)
- The 2 principal components accounted for 62% of the variance in the dataset



K-NEAREST NEIGHBOURS

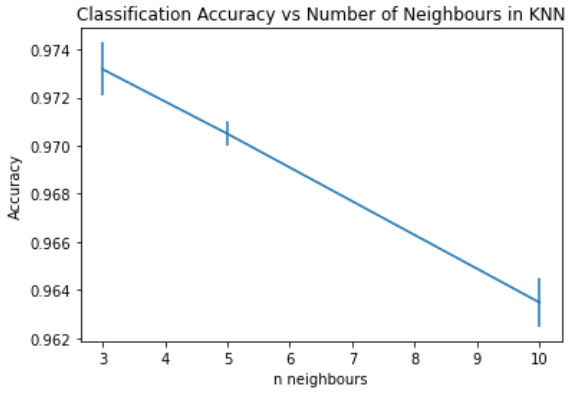
K-Nearest Neighbours

- Instance-based
  - Stores instances of the training data instead of creating a general model
  - Classification is computed from a simple majority vote of the k nearest neighbors of a query instance
- Parameters
  - k – the number of neighbours
  - Weights – uniform, distance, or custom
- Pipeline: PCA followed by KNeighborsClassifier(k)
  - GridSearch ran to find the optimal k

KNN Tuning - Grid Search

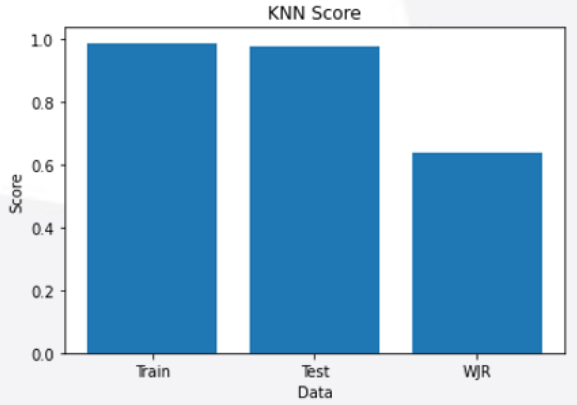
- We decided to use the default 'uniform' weights function
- Ran GridSearch to find the optimal k value between [3, 5, 10]
  - With 3 Cross Validation splits

rank	k	mean_fit_time	mean_score_time	mean_test_score
1	3	500.526219	1.326943	0.973175
2	5	505.309794	1.664336	0.970493
3	10	494.674558	1.785808	0.963481



KNN Testing

- The training and test data scores were very high, as the model was trained with images from all the stations being scored
  - Guaranteed there were similar instances stored to compare against
  - `Train score 0.987246458656878`
  - `Test score 0.9774138904573687`
- Testing with a station (WJR) that the model never trained on resulted in much poorer results*
  - `Test WJR score 0.6401673640167364`
  - * more on the reasons why later



SUPPORT VECTOR MACHINES

Support Vector Machines (SVM)

- Data can be transformed into n dimensions
- Parameters:
  - Kernel: Determines how many dimensions data is projected into
  - C: Determines if the model is learning or memorizing
  - Gamma: Determines the influence of a single training point

- Pipeline: StandardScalar followed by PCA and SVC(C, gamma)
  - The algorithms in scikit-learn are sensitive to scaling, so it is recommended to scale the data
  - GridSearch ran to find the optimal C and gamma

SVM: Kernel

- Kernels to choose from:
  - Radial Basis Function (RBF): data projected in infinite dimensions
  - Polynomial: data projected in degree dimensions
  - Linear: data is linearly separable

- RBF was chosen as the estimator
- Why may this be the case?
  - Snow in images would not always be linearly separable

SVM: C (Regularization parameter)

- Values to choose from:
  - Strictly positive - [0.1, 10] - [small, large]

- 10 was chosen by GridSearch
- Why may this be the case?
  - The low C of 0.1 would likely be too simple of a model to classify without huge number of inaccuracies

SVM: Gamma (Kernel coefficient)

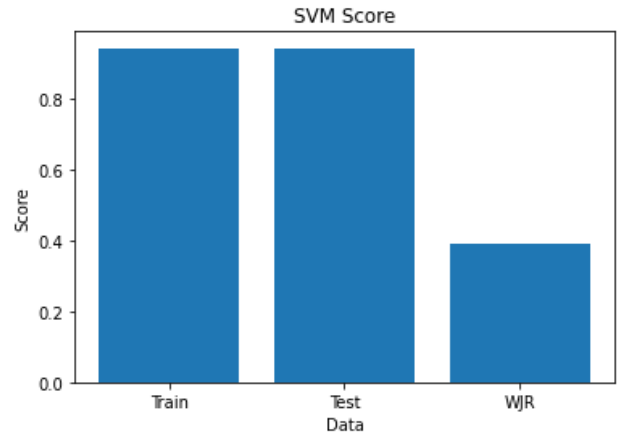
- Values to choose from:
  - Strictly positive - [0.1, 1] - [small, large]

- 0.1 was chosen by GridSearch
- Why may this be the case?
  - The larger gamma of 1 means that the radius of each training point is small
  - Snow might be in large patches, sides of the image, flurries over the entire image

SVM Testing

- Train and test had similar performance because all the stations in the training data were seen
- WJR was an unseen station and it performed worse on that

```
Train score 0.9416443126735375
Test score 0.9408996800301148
Test WJR score 0.3891213389121339
```



RANDOM FOREST – DECISION TREE

Random Forest – Decision Trees

- A supervised learning algorithm that fits multiple decision tree classifiers and uses averaging to improve the predictive accuracy and control over-fitting.
- Parameters:
  - n_estimators – number of trees in the forest
  - max_features – number of features to consider for the best split
  - max_depth – maximum depth of each tree
  - criterion – function to measure the quality of a split
- Our Pipeline: PCA followed by RandomForestClassifier()
  - Grid Search ran to find an optimal set of parameters

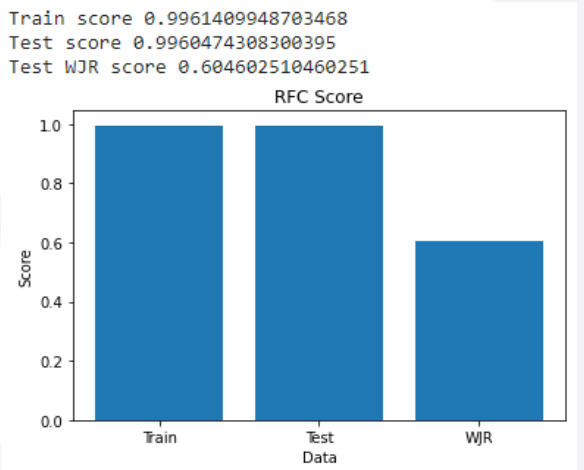
Random Forest Tuning - Grid Search

- Ran Grid Search to find an optimal set of parameter values
- Cross validation splits: 3
- Parameter grid used:
  - n_estimators: 50, 100, 250
  - max_features: auto, log2
  - max_depth: 5, 10, 25
  - criterion: gini, entropy
- Grid Search took ~12-14 hours to run all parameter permutations
- Optimal parameters found:
  - n_estimators: 250
  - max_features: auto
  - max_depth: 25
  - criterion: entropy

Random Forest Testing

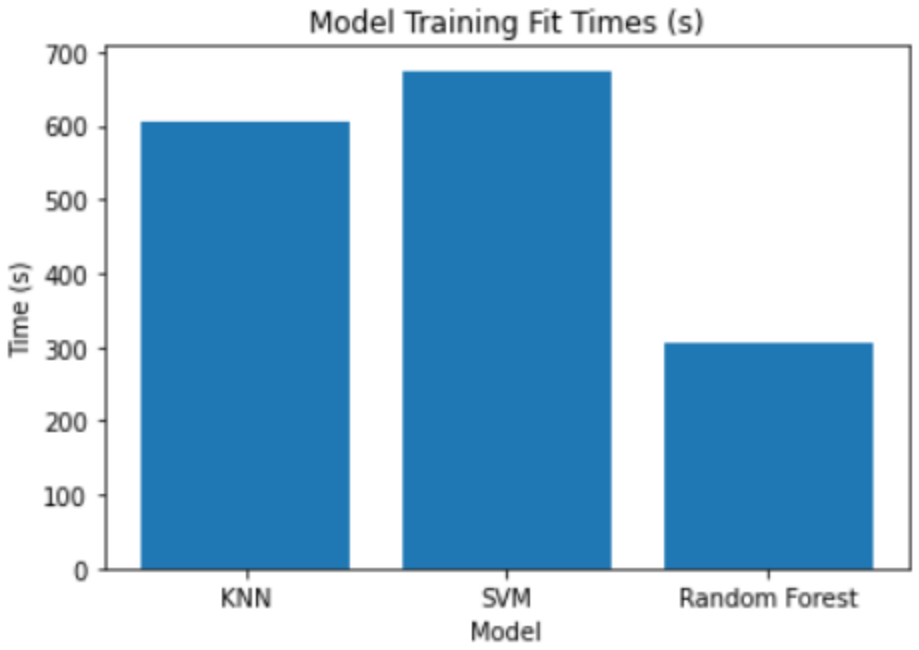
- Train and test had similar/high performance because all the stations in the training data were seen
  - `Train score: 0.99614099487`
  - `Test score:  0.99604743083`
- Data from station WJR was never used in the initial train/test of the model and had performed worse*
  - `Test WJR score: 0.60460251046`

*more on the reasons why in the lessons learned



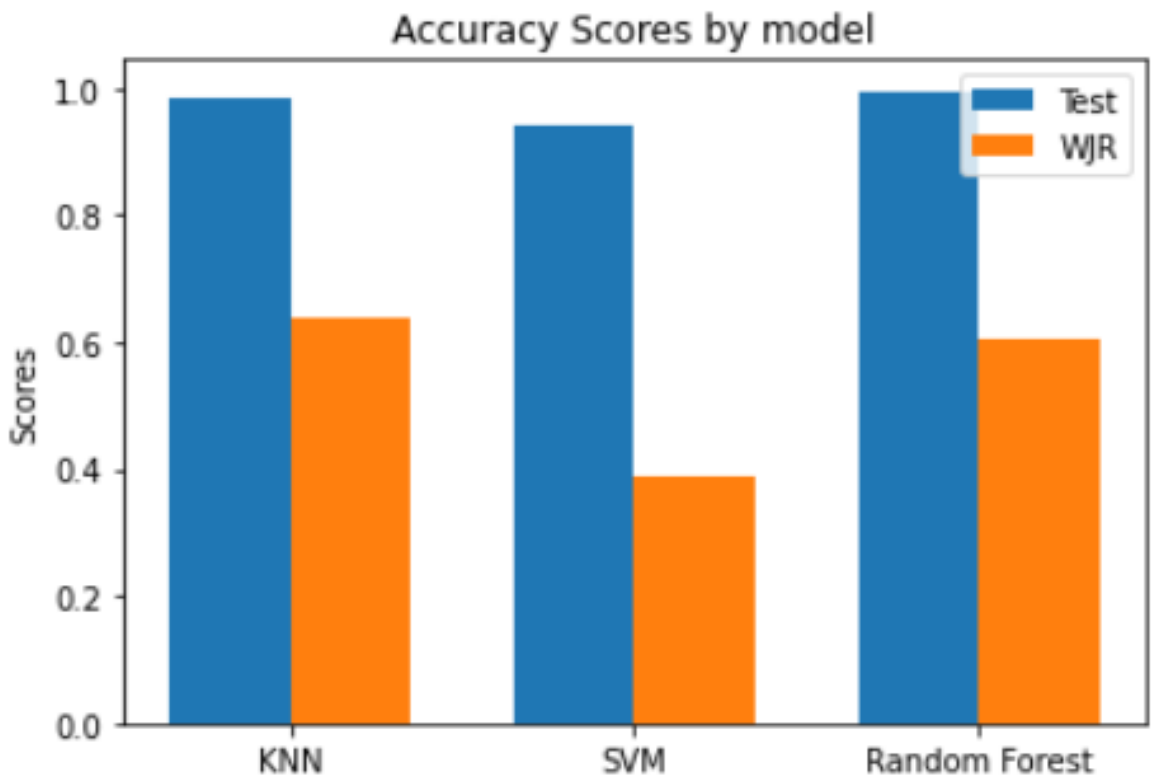
COMPARING THE MODELS

Training Time



- Shown here are the training fit times with the best parameters found via gridsearch
- Random Forest had a significantly shorter training time

Accuracy

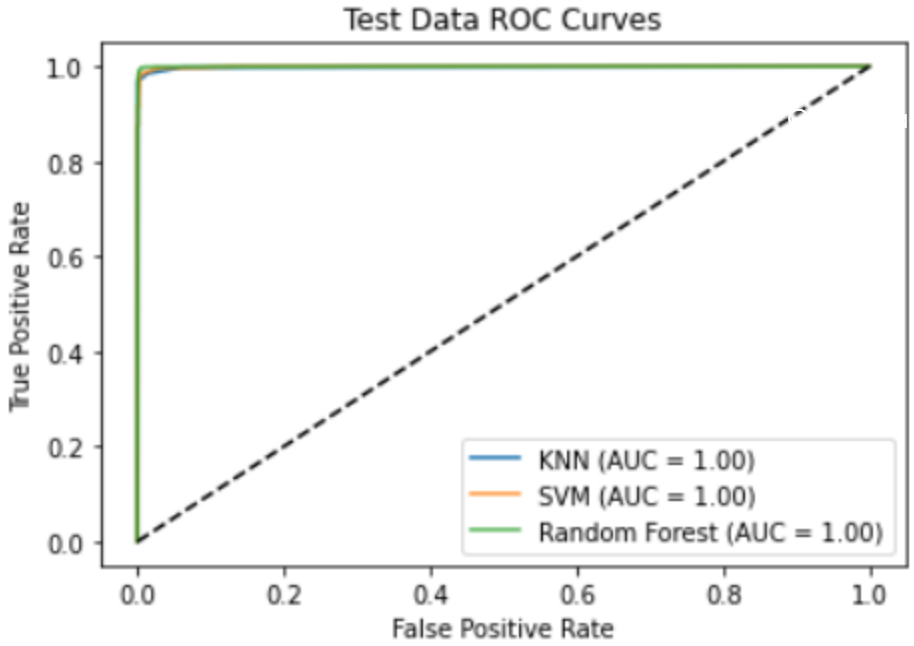


- All the models performed well (0.94-0.97) on the test data
- They performed poorly on WJR
- KNN yielded a marginally better result in both test datasets

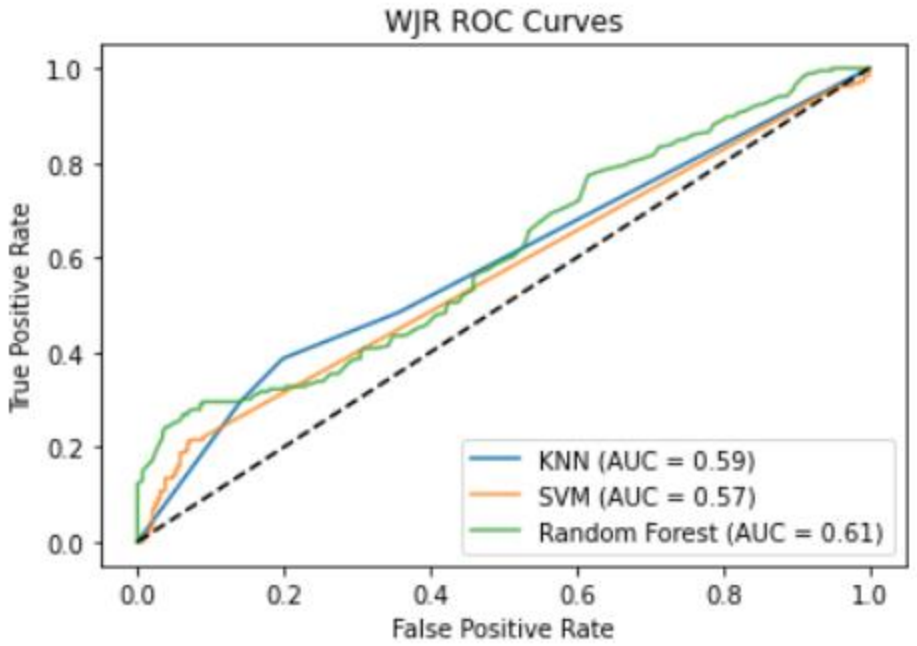
ROC Curves

Area under the receiver operating characteristic (ROC) curve

All 3 models had similar great performance on the test dataset



With WJR Random Forest yielded a slightly better result than the others



CONCLUSION AND LESSONS LEARNED

Station WJR (Excluded from training)

- The low accuracy for WJR (~60%) was due to our incorrect labelling
- The model was making correct predictions but did not match the target label

Conclusions

- Given the comparably shorter training time and high accuracy AUROC score Random Forest seems to be the optimal choice
- Dimensionality Reduction (PCA was used) greatly reduced the training time
  - SVM training without PCA took over 4 hours (before giving up). With PCA it was reduced to 106 minutes (over half the time)
- The model performed very well when the test data was for stations that were part of the training but poor for the never-before-seen station (WJR)
  - Though this may have been mostly due to the mislabeling

Lessons Learned

- The images are not always reliable at determining the condition of the snow sensor
- To ensure the accuracy and reliability of the images we should only use images that are pointing at the sensor
  - This would only apply to the Snow On Ground sensor
- Correct labeling of the data is critical in producing an accurate model
  - If we use observation data to label images, we should ensure the data was valid
  - Manual labeling or ML clustering the images could help this
  - We may need to re-evaluate the threshold for labeling 'Snow'

Questions?