Legal & Machine Learning Documentation

Legal & Machine learning Documentation - Improve the Road Safety in Breda

Ron L. Tabuchov
Mohamed K. A. M. Elshami
Daria Vlăduţu
Peter Husen

Data Science and Artificial Intelligence, Breda University of Applied Science

Index

1 Background

This document outlines architecture of models used within an AI system developed to improve road safety by classifying and alerting users the risk level of each road. This documentation is compliant with regulations set out by the EU AI act, and therefore comprises the model architecture, intended use case, and a step-by-step walkthrough of the system.

2 Goal

The goal of this file is to document the steps taken to build the system for road safety, this is both to fulfil legal obligations as well as to inform users of how models operate, and therefore provide assurance that the model will help them drive more safely.

3 Method

Model Selection: Choose the appropriate ML and deep learning algorithms based on the problem and data characteristics.

Model Architecture: Define the architecture of the deep learning model, including the number of layers, neurons, activation functions, etc.

Hyperparameter Tuning: Set initial hyperparameters and use techniques like grid search or random search to find the optimal values.

Model Training: Train the model using the training dataset, monitoring the training process for convergence and performance.

Validation: Validate the model on the validation dataset to tune hyperparameters and prevent overfitting.

Performance Metrics: Evaluate the model using relevant metrics (e.g., accuracy, precision, recall, F1-score).

Review and Follow-Up: Regularly review the developed ML in GitHub and the progress made on the actionable steps, adjusting the code and methods as needed to ensure continuous improvement.

4 EU Artificial Intelligence Act

Article 5:
Our model may not subvert people’s decision making, it should offer people the option of taking a safer route but not force them to do this. We also can't take advantage of people’s vulnerabilities (we aren't planning on either of these).

Article 6:
Our AI system is intended to be used as a safety component (safer driving)
It does use third-party as we aren't directly related to ANWB (yet)
Therefore, the system is high risk.

Article 7:
If the commission were to assess our AI system
A: Our intended purpose is to improve road safety by warning people about dangerous roads and choosing safer options or driving more carefully in those areas.
B: The AI system would likely be used by those who already opt to drive more safely
C: Personal data is removed for the purpose of assessing road safety.
D: The AI system only informs and doesn't make any decisions by itself. Recommendations shouldn't lead to more harm, but if users drive more carelessly due to not receiving a notification of approaching a dangerous road this could lead to harm.
E: The AI system is new so hasn't already caused harm.
F: see E
G: Opting out from outcomes is practical as the driver can still choose the road themselves.
H: The users of the AI system are the drivers; they are not put in a vulnerable position by our ML model.
I: If drivers approach a high-risk road, they can still opt to change their route by taking a turn somewhere so the decision is corrigible, however this might be a problem if hastily taken a corner.
J: The product is likely to improve road safety.

Article 9:
1: we must establish a risk management system, and document what we do with it.
2: identify reasonably foreseeable risks that can be seen when using the system; estimate risk of using, as well as misusing the app (misuse would probably be irrelevant, unless people drive less safe in high-risk zones),
We can do this by identifying false positives/negatives or false classifications of roads in our case and try to mitigate the amount.
We will address the likelihood and impact of the risks
For long-term we would develop strategies to minimise the risks by keeping algorithms up to date and making sure data quality is as high as possible.

Article 10:
Required to have good quality datasets for training validation and testing
We can ensure we have this by
Having high-quality data representing road safety (ANWB dataset is representative of the data)
Data should be relevant to road conditions (The data is categorised by road)
Have diverse data to avoid bias (Data is from all over Breda)
Implement processes to ensure data integrity and secure handling (Password login to access data, columns that seem wrong aren't used).

Article 11:
Create and maintain comprehensive technical documentation
We should document our architecture, algorithms and our source in a separate file.
We should state our intended purpose (Improving road safety by classifying roads as dangerous and informing end users of these roads)
Include performance metrics of the models
Maintain documentation on improvements (markdowns on models and changes).

Article 12:
Keep logs to monitor the system operations (this would only be relevant post deployment; therefore, it is irrelevant for the scope of our project.
Logs of outputs are still relevant so we should keep those.

Article 13:
Provide clear information on ai to users
Provide detailed instructions on how the AI works
Describe the limitations of the system
Offer a way to contact services to assist.

Article 14:
Ensure human oversight is in place to mitigate risk
Humans should be able to override the decisions the model makes
Staff would have to be trained to intervene (Not relevant yet in our case, since we won't use the model for real-time cases yet.
Continuously monitor the performance and involve humans in critical decision making (Continuous monitoring is relevant during and after deployment not before).

Article 15:
Ensure systems are accurate and secure
Test the model with different conditions
Update the model to improve performance and address new data (new data might not be relevant for the scope of our project)
Implement cybersecurity measures to protect the data (Password and vpn protection on the data).

Article 16:
Implement quality management for the model
Develop a procedure to ensure quality remains high
Conduct external audits to ensure compliance (we can ask fellow students or perhaps the teachers to check if they feel our model is compliant)
Make sure we all try to help each other improve by integrating feedback.

Article 17:
Document the quality management system with our policies and objectives
Document the procedures for design
Meet quality management standards
Improve based on received feedback (limited in scope due to our project scope).

Article 18:
Maintain documents after placing on the market (we won't actually place it so this is irrelevant)
Update the documents (irrelevant for the same reason)
Accessible to relevant authorities (irrelevant for the same reason).

Article 19:
Automatically generate logs (Output is documented earlier, new data won't be input during the scope of the project.)
Logs must be retained for a sufficient period to enable monitoring. (Should we log things we can maintain them, but we likely won't add new events.)
Logs must be integral to prevent tampering. (See before)

Article 20:
If a large problem arises, we have the duty to pull the plug, and take appropriate measures to get the model back on track. (Since our model is more of a proof of concept than an actual product this will likely not be relevant).

Article 21:
Should a relevant authority ask for information and documentation we will provide it, as well as the automatically generated logs. This will be confidentially shared. (This is unlikely to happen, but if it will we should be ready).

Article 22:
Article 22 references launching AI systems from outside the EU in the EU, since we are in the EU this is irrelevant for us.

Article 23:
Article 23 talks about an importer, if we were to sell this product to ANWB they would be the importer therefore this is not relevant for us.

Article 24:
Article 24 talks about the distributor which would also be ANWB not us who has to verify we adhere to standards.

Article 25:
Article 25 states that everyone involved within the AI is responsible for the legal distribution of the AI which would relate to us. Other parts involve handing the AI system over the creators have to make sure the new owners have the necessary knowledge to keep working on it. This would be relevant only if ANWB wishes to take over our models.

Article 26:
Organisational oversight must be present for high-risk AI systems as well as instructions to use it, this must be done by trained individuals.
Data must remain relevant for the intended purpose, and continuously monitored.
Employers must know they work with AI, and public authorities must comply with registration obligations.
People affected by AI decision making should be informed of this.
All these points are also covered in other points, so it doubles as a bit of a summary too.

Article 27:
We need to describe how the AI system will be used, during which time and the frequency of usage, categories of natural persons likely to be affected (no one, it is an opt in system), specific harm it might have on people (people potentially driving less safe because they are in a "low risk" zone)
Articles 28-39, part of section 4, are all about authorities that check whether AI systems are compliant with the rules. They do not pertain to actions we have to consider to be within legal parameters.

Article 40:
Should a request from the European Commission come in we would have to follow their standardisation request, we would have to adhere to published standards. It is also requested that our models are as efficient as possible, so no computational power is wasted.

Article 41:
Article 41 addresses the adherence to standard rules, or where there is a lack of there is a requirement to prove adherence to the rules in other ways. If we meet the created standards, which we should if following the previous points, then this isn't important.

Article 42:
This article says that if an AI system has been trained and tested on specific settings (like our model being trained on specifically Breda), then it is compliant with the rules in article 10. It also mentions that if we have a cybersecurity statement of conformity then we comply with the requirements of article 15. If we wish to deploy the model for ANWB we would have to get a statement of conformity.

Article 43:
Article 43 discusses the processes of assessing if our model meets the correct standards, and how we would have to go about having our model assessed to meet the required standards. Therefore, if we follow the steps from earlier we should be fine.

Article 44:
Article 44 mentions the specifics of certificates and are not relevant for us. What is relevant is that if our model is deemed to no longer meet the requirements then they can ask us to change the model.

Article 45:
Mentions the obligations of notified bodies, this is what they must do after we tell them what we have done and is not relevant for us.

Article 46:
This mentions exceptions to the rule in the case of public safety or environmental protection concerns. Our model does not apply to these and therefore this rule is not applicable to our situation.

Article 47:
We need a written document of conformity for each of the systems mentioned previously. This document must be kept available for 10 years, as well as be in a language that can be understood (English should suffice).

Article 48:
AI systems are subject to CE law; therefore, a CE mark would have to be present somewhere in documentation where it is easily accessible.

Article 49:
Before putting our model on the market, we must register in the EU database, if we work with ANWB they likely have an authorised representative who would do this.

Article 50:
We must inform users that they are dealing with an AI model. This can be done by having this mentioned explicitly before users use the app.

Article 51:
This is a commission classification based on impact capability and computational power. We do not require as much computational power as is mentioned here for our preliminary model.

Article 52:
If we do end up meeting the requirements set by article 51, we will have to notify the commission. They will then investigate further.

Article 53:
This links back to keeping up to date logs of our model including training and testing processes and evaluations. This part mentions that we have to share this information with the commission if our system gets large enough to be investigated under article 51

Article 54:
Article 54 is for representatives of companies from outside the EU. We are inside the EU, so this is not applicable to us.

Article 55:
This article mentions following the standard protocols set earlier in articles 24 and 40.

Article 56:
The EU AI office is drafting rules for how to use AI, these rules are not yet implemented and should be within a year of the act going into effect. Once these come into practice we should adhere to them. This is not yet relevant for us.

Article 57:
There should be an AI sandbox on national level where the model can be tested before being deployed, if this is satisfied our model is fine. This is however not an action we necessarily have to take but more of a check if we indeed fit all the requirements

Article 58:
Same sandbox as 57, not our responsibility.

Article 59:
Same sandbox as 57, not our responsibility.

Article 60:
The testing of AI models in real world instances must be submitted to the AI office to receive a permit for testing. The testing must be done within 6 months unless an extension is approved.

Article 61:
When/if we test our model, we will have to have explicit consent from everyone involved in the test, this consent must be dated and documented as well as a copy given to the person who is partaking in the test.

Article 62:
As mentioned earlier there are standardised templates being created as well as sandboxes developed where testing can take place. SMEs (small and medium-sized enterprises) have priority access to these so they can get started. Should we wish to deploy these models we would have to contact them to see the availability of these.

Article 63:
This mentions that a microbusiness that is not linked to a larger company is exempt from several quality management systems. We would be linked to ANWB, so this is not relevant to us.

5 Legal Requirements and Obligations

To comply with legal requirements the articles we have examined are articles 5-63, most of these articles provide relevant information for the providers of AI systems, which would be us, and some provide information for other parties. This section will first detail the articles which mentions things providers have to comply with, and then mention which articles do not require further attention, either at all or for the scope of this project.

Articles 5-17, 19-21, 25-27, 40, 47-50, 52, 60-63 are relevant for our project. Some of these are steps that are to be taken during the scope of this project, others would be relevant in a future stage of deployment. To be compliant with these rules we must

- Make sure we don't subvert the ability to make decisions or take advantage of vulnerabilities.
- Implement a risk management system.
- Ensure our data is representative of what we are examining, and that this data is relevant. Make sure our data is diverse to avoid bias and implement cybersecurity measures.
- Document our architecture algorithms and source in a separate file.
- State the intended purpose of our model and include performance metrics and document improvements.
- Keep logs of performance after deployment.
- Ensure transparency to users about them using AI as well as how the AI works and in which ways it is limited. Also provide a way for them to contact a person for assistance.
- Allow humans to override AI decisions in places it is necessary.
- Test the model on different systems to ensure the model is accurate and secure.
- Implement quality management for the model and conduct external audits to ensure compliance. Integrate user feedback into the project
- Generate output logs and save these logs to enable monitoring.
- Allow our documentation and logs to be accessible to relevant authorities upon inquiry.
- Describe how the system will be used, and whether or not natural persons will be affected by our AI model.
- Adhere to standardisation requests from the EU if they arrive at a later point in time.
- Provide a written document of conformity.
- Acquire a CE mark and make sure it is easily accessible within our documentation.
- Register in the EU database of AI systems before deploying the model on the market.
- Inform users of our system that they are using an AI model.
- Should we wish to test our model later in real-time we have to submit a request for a permit.
- If we test our model in this way, we need explicit consent from everyone involved.
- Standardised templates are being created, as well as sandbox environments where systems can be tested. We could contact these services once available.

Article 18 is about the documentation of the previously mentioned articles and keeping those up to date, after placing the system on the market. Our system will not actually go on the market yet during this project therefore it is not yet relevant.

Articles 22-24 are for providers from outside the EU, we are not from outside the EU, so this does not apply to us.

Articles 28-39 are all for authorities that check whether AI systems are in compliance with the regulations. If we claim to adhere to other rules, then these authorities follow the rules mentioned within these articles to ensure that we do.

Articles 41-46 discuss the specifics of certain certificates that legal authorities can grant as proof that AI systems are compliant with rules, as well as discussing the processes on how to prove that examined AI models are compliant. These are relevant for authorities examining our models and not for us.

Article 51 mentions special requirements for computationally intensive, or systemically risky AI models. We do not qualify for this. Article 52 provides procedures that AI providers have to take, which would only be relevant if we develop much further on the scope of the project.

Articles 53-56 link back to articles mentioned prior to this, adhering to other articles means adherence to these articles as well.

Articles 57-59 are about nations within the EU providing resources, we can make use of these once they have been set up but are not actions we have to take.

6 Data collection and Usage

Our data has been collected from several sources. First is a dataset provided to us by ANWB, this dataset takes personal information as well as data collected by their behaviour on the road. ANWB only collects this data from members that sign up to their safe driving insurance, which is a higher tier which is optional. Consent into data collection is therefore present. For the purposes of machine learning, all personal data has been removed from the dataset and only an event id is used.

The second dataset that has been used is the KNMI dataset regarding precipitation. This data is freely available from the KNMI website. The data on precipitation is collected based on the water that falls in specific weather stations which measure the amount of rain that falls in that location. For our purposes we have used the nearest weather station to Breda for analysis.

The third dataset we used is the Open-meteo dataset, which is an open-source system that also looks at the weather similarly to KNMI. Open-meteo has 80 years of data meaning that analysis based on it is reliable for the purposes of machine learning.

The preprocessing steps of both models are present in the Preprocessing document.

Using this data, we aim to detect which roads have a disproportionate number of accidents and mark these as high-risk zones. The goal of marking roads as such is to inform people of the increased risk, get them to drive safer or choose an alternative route. By doing this the number of accidents occurring in a year should decrease.

7 System Usage

The system is intended to be used before or whilst driving using a navigation system, navigation systems give the option to drive the fastest route, our model will also give an option for the least risky route. This can be done before beginning with driving but will also offer to change whilst driving. When a user approaches a dangerous road, the system will mention this as well as offering a safer alternative. Users can choose to follow what our system suggest, opt to use the standard navigation, or follow their own road. Our system makes clear to users before usage that it is an AI system, that AI can mistakes, and user discretion is advised. Users should drive with care even if they enter low risk zones, accidents can still occur even if the risk is low.

8 Addressing Legal Obligations

Our AI system, developed to improve road safety by identifying risky roads, is in full compliance with the EU Artificial Intelligence Act. We have ensured that our system does not subvert users decision-making abilities or exploit their vulnerabilities, referring to Article 5. As a safety component, our system qualifies as high-risk under Article 6, necessitating robust risk management and data quality protocols as specified in Articles 9 and 10. We have implemented a comprehensive risk management system to identify and mitigate foreseeable risks, including false positives and negatives, ensuring continuous improvement and data quality.

Our data is representative, diverse, and securely handled, with all personal data anonymized to protect user privacy. We maintain detailed technical documentation of our model architecture, algorithms, and performance metrics as required by Article 11, and ensure transparency to users about the system's functionality and limitations in line with Article 13. Human oversight is integrated into our system to allow for decision overrides, enhancing safety and compliance with Article 14. Moreover, we have established cybersecurity measures to protect the data and ensure system accuracy and security, fulfilling Article 15 requirements.

Throughout the development and potential deployment phases, we are committed to adhering to quality management standards, conducting external audits, and integrating user feedback as stipulated in Articles 16 and 17. Before market deployment, we will register our system in the EU database and ensure users are informed that they are interacting with an AI model, complying with Articles 18, 19, and 21.

9 Random Forest

Iteration 1

```
df_selected = df_joined[['date', 'eventid', 'duration_seconds',
'road_name', 'maxwaarde', 'incident_severity', 'risk_category_encoded',
'weather_code', 'risk_level_wmo']].copy()
```

Selecting columns and creating a copy to avoid modifying the original DF

```
incident_counts =
df_selected.groupby('road_name').size().reset_index(name='incident_count')
```

grouping the data frame by road name and resets the index of the resulting Series and converts it into a df with a new column named.

```
df_selected = df_selected.merge(incident_counts, on='road_name',
how='left')
```

merging the columns to the selected columns on ‘road_name’ to combine the data of the new column (‘incident count’)

```
encoder = LabelEncoder()
df_selected['road_name'] =
encoder.fit_transform(df_selected['road_name'].astype(str))
```

Label encoder converts the categorical ‘road_name’ column into numerical labels suitable for machine learning algorithms

```
X = df_selected[['road_name', 'weather_code', 'risk_level_wmo',
'incident_count']]  # Features
y = df_selected['risk_category_encoded']  # Target
```

X represents the selected columns to use an input: 'road_name', 'weather_code', 'risk_level_wmo', and 'incident_count'.

Y is the target variable, where 'risk_category_encoded' is used as the target variable that model will try to predict based on the input features.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Splitting the data set to train and test.

```
 plot_class_distribution(y, title):
    counter = Counter(y)
    classes = list(counter.keys())
    counts = list(counter.values())

    plt.figure(figsize=(8, 6))
    sns.barplot(x=classes, y=counts, palette='viridis')
    plt.xlabel('Class')
    plt.ylabel('Frequency')
    plt.title(title)
    plt.show()

plot_class_distribution(y_train, 'Class Distribution Before SMOTE')

smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)


 plot_class_distribution(y, title):
    counter = Counter(y)
    classes = list(counter.keys())
    counts = list(counter.values())

    plt.figure(figsize=(8, 6))
    sns.barplot(x=classes, y=counts, palette='viridis')
    plt.xlabel('Class')
    plt.ylabel('Frequency')
    plt.title(title)
    plt.show()

plot_class_distribution(y_train, 'Class Distribution Before SMOTE')
```

Plotting class distribution before and after SMOTE and using the SMOTE method to, ensure balanced representation of the classes in the target variable column ('risk_category_encoded').

```
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train_resampled, y_train_resampled)
```

Creating and training Random Forest Classifier because of its ability to handle various of data type, balanced and imbalanced data sets. In addition, Random Forest offers benefits such as reducing overfitting, handling missing values, and providing feature importance, making it a suitable choice for classification task.

```
y_pred_test_rf = rf.predict(X_test)
print("RandomForest Test Accuracy for Risk Level Classification:",
accuracy_score(y_test, y_pred_test_rf))
print("RandomForest Test Classification Report for Risk Level
Classification:\n", classification_report(y_test, y_pred_test_rf))
```

Making predictions on the test data and evaluate the predictions by comparing the true labels (y_test) with the predicted labels by the machine learning algorithm (y_pred_test_rf). Generating classification report that includes precision, recall, f1-score, and support for each class.

Precision: The ratio of correctly predicted positive observations to the total predicted positives.

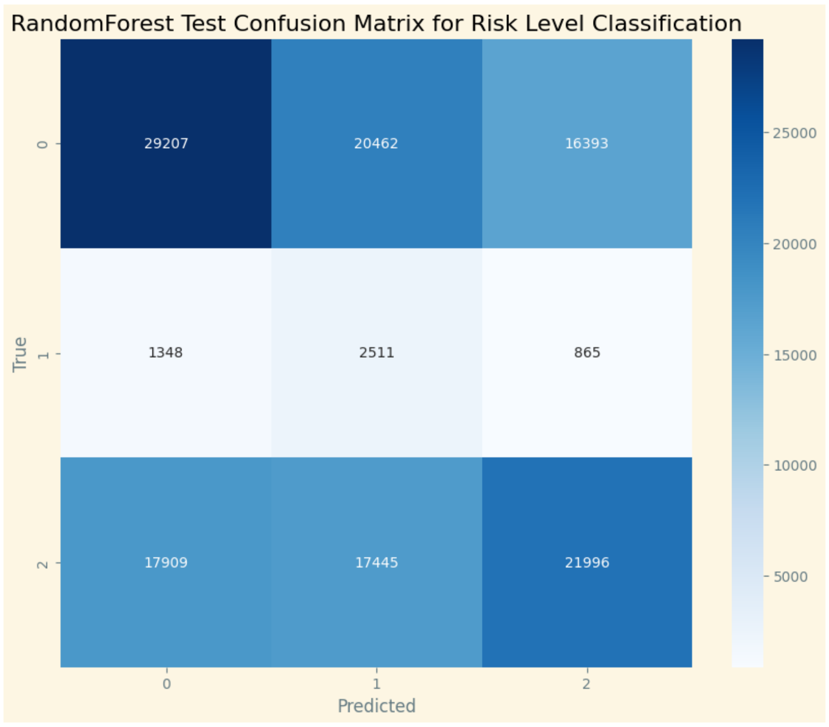
Recall: The ratio of correctly predicted positive observations to all observations in the actual class.

F1-Score: The weighted average of precision and recall. It considers both false positives and false negatives.

Support: The number of actual occurrences of the class in the dataset.

40.37% of the predictions made by the model are correct. This low accuracy indicates that the model may not be performing well on this classification task.

```
def plot_confusion_matrix(y_true, y_pred, title):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title(title)
    plt.show()

plot_confusion_matrix(y_test, y_pred_test_rf, 'RandomForest Test Confusion
Matrix for Risk Level Classification')
```



Plotting confusion matrix and examining the error analysis point out a large proportion of Class 0 and Class 2 instances are being confused with each other and with Class 1. This suggests that the features used aren't capturing the distinctions between these classes. Data preprocessing and better features selecting will provide effective and reliable machine learning model.

Iteration 2

Trying different method to achieve higher accuracy by adding additional preprocessing steps, weights, and fine tuning for the model. In iteration 2, we count the total number of accidents per road, assign weights to each incident type according to its severity and use the machine learning model to classify risk level according to the new selected features. By calculating the total number of incidents per road, we can identify roads with high incident frequencies and potentially investigate the causes or implement measures to reduce incidents.

This line counts the number of occurrences of each unique value in the road_name column. It returns 2 columns where the index is the unique road names and the values are the counts of incidents.

```
incident_counts = df['road_name'].value_counts().reset_index()
incident_counts.columns = ['road_name', 'total_incidents']
```

By calculating the total number of incidents per road, you can identify roads with high incident frequencies and potentially investigate the causes or implement measures to reduce incidents.

```
incident_types_per_road = df.groupby(['road_name',
'incident_severity']).size().unstack(fill_value=0).reset_index()
```

create a data frame that shows the counts of each type of incident severity for each road.

```
road_incident_data = pd.merge(incident_counts,
incident_types_per_road, on='road_name')
```

This code merges two data frames, incident_counts and incident_types_per_road, based on the road_name column.

```
severity_weights = {
    'HA1': 1, 'HA2': 2, 'HA3': 3, 'HB1': 1, 'HB2': 2, 'HB3': 3, 'HC1': 1,
'HC2': 2, 'HC3': 3,
    'HC4': 2, 'HC5': 3, 'HC6': 4, 'HC7': 3, 'HC8': 4, 'HC9': 5, 'HC10': 4,
'HC11': 5, 'HC12': 6,
    'HC13': 2, 'HC14': 3, 'HC15': 4, 'HC16': 3, 'HC17': 4, 'HC18': 5,
'HC19': 4, 'HC20': 5, 'HC21': 5
}
```

Defines a dictionary called severity_weights which assigns a weight to each type of incident severity. The weights indicate the severity of each type of incident, with higher weights representing more severe incidents.

```
      for severity, weight in severity_weights.items():
   if severity in road_incident_data.columns:
         road_incident_data[f"{severity}_weighted"] =
      road_incident_data[severity] * weight
```

Iterates through the severity_weights dictionary and applies the weights to the corresponding columns in the road_incident_data data frame to calculate the weighted counts for each incident severity type.

```
      incident_columns = [col for col in road_incident_data.columns if
      'weighted' in col]
      road_incident_data['severity_score'] =
      road_incident_data[incident_columns].sum(axis=1)
```

This code snippet selects the columns that contain weighted incident counts and calculates a severity_score for each row in the road_incident_data data frame by summing these weighted counts.

```
q25 = road_incident_data['severity_score'].quantile(0.25)
q75 = road_incident_data['severity_score'].quantile(0.75)

bins = [road_incident_data['severity_score'].min() - 1, q25, q75,
road_incident_data['severity_score'].max()]
labels = ['low', 'mid', 'high']
```

This code snippet categorizes the severity_score for each road into risk levels (low – min score to first 25%, mid – 25% to 75% , high – 75% and above) based on predefined bins. It uses the pd.cut function from pandas to achieve this.

```
road_incident_data['risk_level'] =
pd.cut(road_incident_data['severity_score'], bins=bins, labels=labels)
```

This code sorts the road_incident_data data frame by the severity_score.

```
X = road_incident_data.drop(columns=['road_name', 'total_incidents',
'risk_level', 'severity_score', 'HA1', 'HB1', 'HC1', 'HC2', 'HC3', 'HC4',
'HC5', 'SP1',
    'HA2', 'HA3', 'HB2', 'HB3', 'HC6', 'HC7', 'HC8', 'HC10', 'HC13',
'HC14', 'SP2', 'SP3',
    'HC15', 'HC16', 'HC17', 'HC19', 'SP4', 'SP5'])

y = road_incident_data['risk_level']
```

This code prepares the feature X and the target y for training a machine learning model, as it selects the relevant features and the target variable.
Dropping 'road_name', 'total_incidents', 'risk_level' columns ensures that the model is trained on relevant, non-redundant features while avoiding data leakage and overfitting. This leads to a more accurate and generalizable model when predicting the risk level of roads based on incident data.

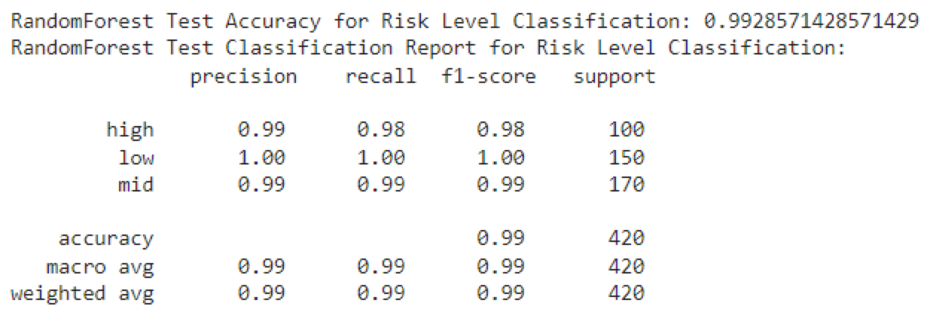
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

Splitting the dataset into training and testing sets. The training set is used to train the machine learning model, and the testing set is used to evaluate the model's performance on unseen data.

```
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
```

This code snippet trains a RandomForestClassifier using the training dataset.

```
y_pred_test_rf = rf.predict(X_test)
print("RandomForest Test Accuracy for Risk Level Classification:",
accuracy_score(y_test, y_pred_test_rf))
print("RandomForest Test Classification Report for Risk Level
Classification:\n", classification_report(y_test, y_pred_test_rf))
```



```
RandomForest Test Accuracy for Risk Level Classification: 0.9928571428571429
RandomForest Test Classification Report for Risk Level Classification:
              precision    recall  f1-score   support

        high       0.99      0.98      0.98       100
         low       1.00      1.00      1.00       150
         mid       0.99      0.99      0.99       170

    accuracy                           0.99       420
   macro avg       0.99      0.99      0.99       420
weighted avg       0.99      0.99      0.99       420
```

Evaluates the trained RandomForestClassifier model on the test dataset by predicting the target values and calculating performance metrics such as accuracy and classification report.\

Iteration 3

```
X = road_incident_data.drop(columns=['road_name', 'total_incidents',
'risk_level', 'severity_score', 'HA1', 'HB1', 'HC1', 'HC2', 'HC3', 'HC4',
'HC5', 'SP1',
    'HA2', 'HA3', 'HB2', 'HB3', 'HC6', 'HC7', 'HC8', 'HC10', 'HC13',
'HC14', 'SP2', 'SP3',
    'HC15', 'HC16', 'HC17', 'HC19', 'SP4', 'SP5'])

y = road_incident_data['risk_level']
```

This code prepares the feature X and the target y for training a machine learning model, as it selects the relevant features and the target variable.
Dropping 'road_name', 'total_incidents', 'risk_level' columns ensures that the model is trained on relevant, non-redundant features while avoiding data leakage and overfitting. This leads to a more accurate and generalizable model when predicting the risk level of roads based on incident data.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

Splitting the dataset into training and testing sets. The training set is used to train the machine learning model, and the testing set is used to evaluate the model's performance on unseen data.

```
min_samples = class_distribution.min()
```

Find the class with the smallest number of samples.

```
k_neighbors = min(5, min_samples - 1)
```

This ensures that the number of neighbors used for generating synthetic samples is at least 1 and at most 4, depending on the smallest class size. This prevents k_neighbors from being larger than the available samples in the smallest class.

```
smote = SMOTE(random_state=42, k_neighbors=k_neighbors)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
```

Initializing the SMOTE object with the specified number of neighbors and a random state for reproducibility. It then fits SMOTE to the training data and generates the resampled dataset.

```
# Check the class distribution after SMOTE
class_distribution_resampled = y_resampled.value_counts()
```

Check the class distribution after SMOTE.

```
X_train_res, X_test_res, y_train_res, y_test_res =
train_test_split(X_resampled, y_resampled, test_size=0.4, random_state=42)
```

Split the resampled dataset.

```
rf_res = RandomForestClassifier(random_state=42)
rf_res.fit(X_train_res, y_train_res)
```

This code snippet trains a RandomForestClassifier using the training dataset.

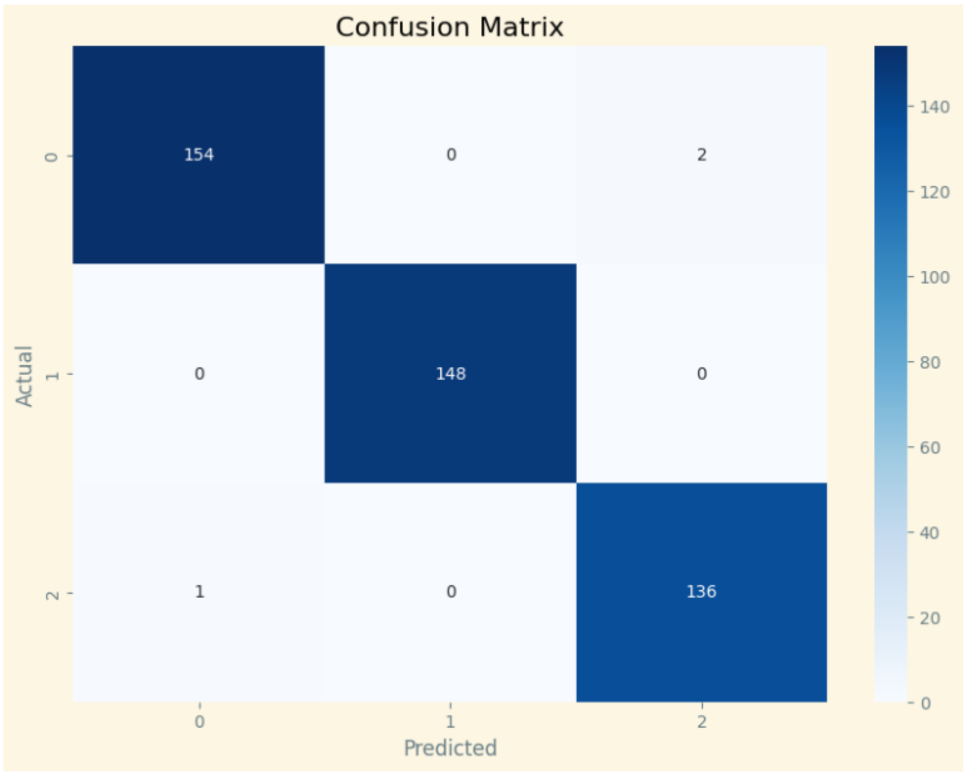
```
y_pred_test_res_rf = rf_res.predict(X_test_res)
print("RandomForest Test Accuracy for Risk Level Classification:",
accuracy_score(y_test_res, y_pred_test_res_rf))
print(classification_report(y_test_res, y_pred_test_res_rf))
```

Evaluates the trained RandomForestClassifier model on the test dataset by predicting the target values and calculating performance metrics such as accuracy and classification report.

```
RandomForest Test Accuracy for Risk Level Classification: 0.9931972789115646
              precision    recall  f1-score   support

        high       0.99      0.99      0.99       156
         low       1.00      1.00      1.00       148
         mid       0.99      0.99      0.99       137

    accuracy                           0.99       441
   macro avg       0.99      0.99      0.99       441
weighted avg       0.99      0.99      0.99       441
```

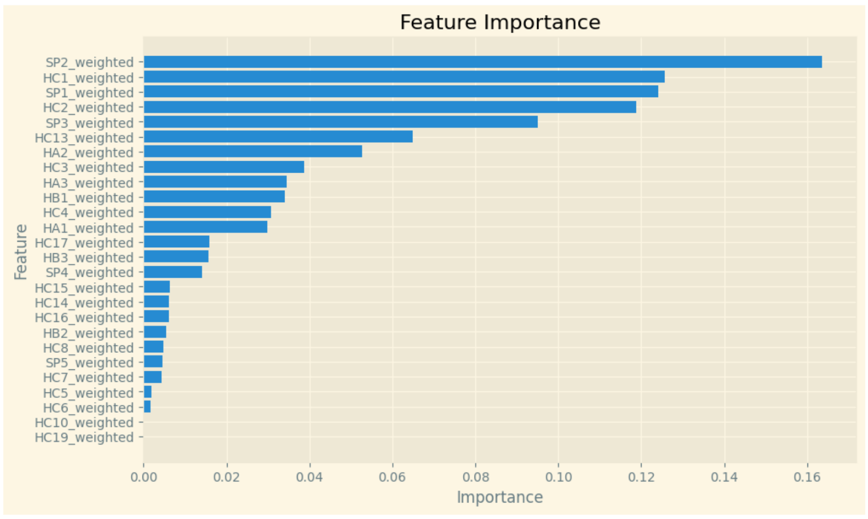
```
conf_matrix = confusion_matrix(y_test_res, y_pred_test_res_rf)
plt.figure(figsize=(10, 7))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Plotting confusion matrix



10 XAI & Interpretability - Random Forest Model

Feature importance:

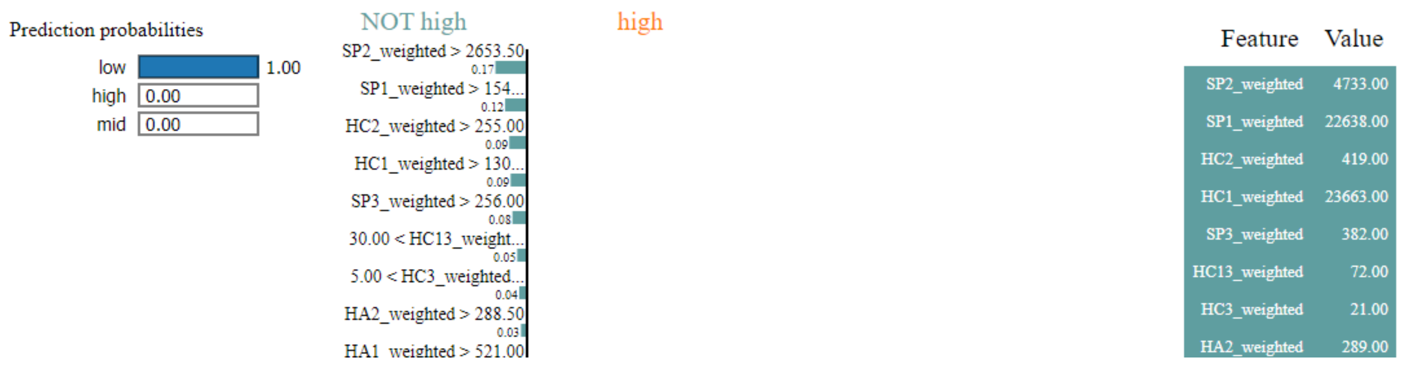


LIME Plot:

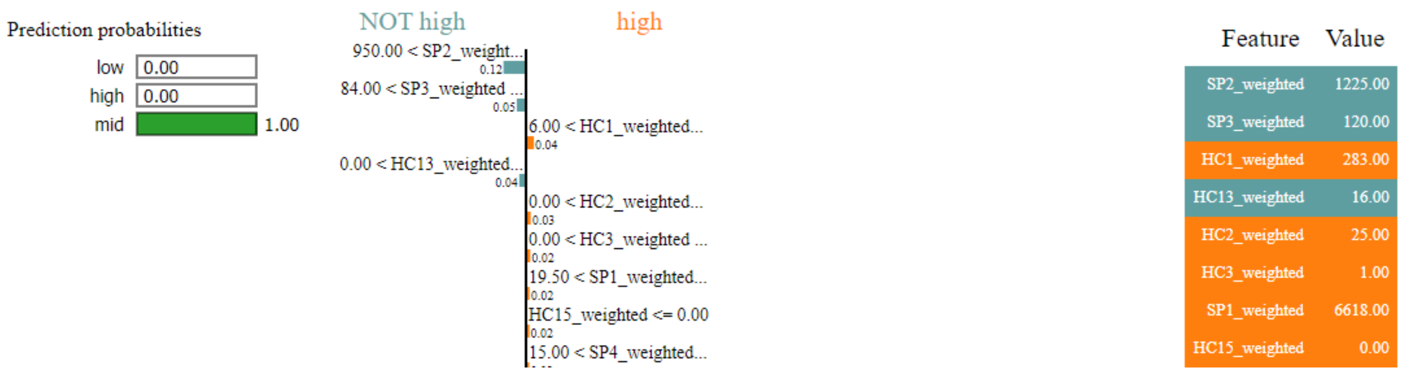
The LIME explanation plot provides insights into how the model made its prediction for a specific instance. The plot displays the prediction probabilities for each class and highlights the contributions of each feature to the model's prediction.

Features are listed along with their contributions to the prediction. The bars indicate how much each feature pushed the prediction towards a specific class.

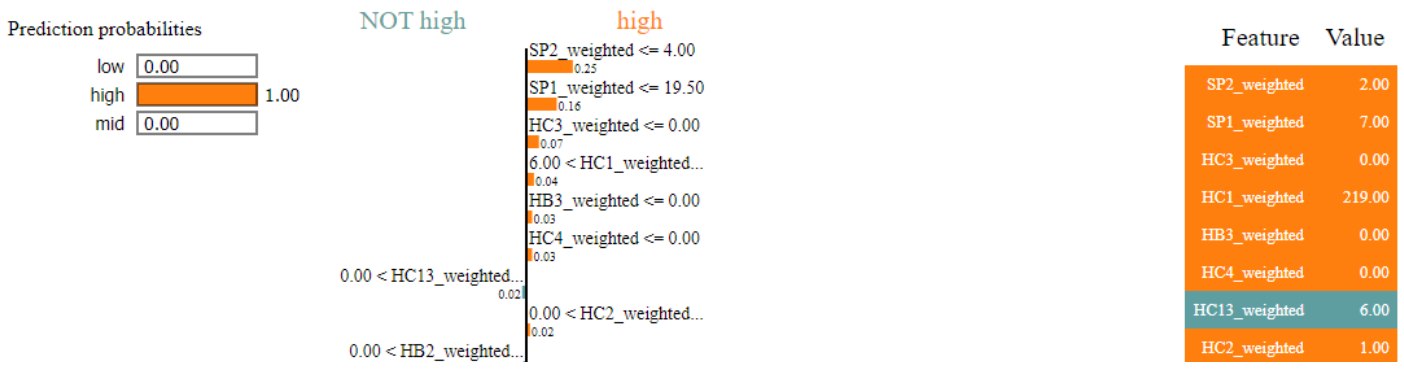
```
explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data=X_train_res.values,
    feature_names=X_train_res.columns,
    class_names=y_train_res.unique(),
    mode='classification'
)

def predict_fn(x):
    x_df = pd.DataFrame(x, columns=X_train_res.columns)
    return rf_res.predict_proba(x_df)

i = 1                            # index of the instance to explain
exp = explainer.explain_instance(
    data_row=X_test_res.iloc[i],
    predict_fn=predict_fn
)

exp.show_in_notebook(show_all=False)
```



Feature	Value
SP2_weighted	4733.00
SP1_weighted	22638.00
HC2_weighted	419.00
HC1_weighted	23663.00
SP3_weighted	382.00
HC13_weighted	72.00
HC3_weighted	21.00
HA2_weighted	289.00



Feature	Value
SP2_weighted	1225.00
SP3_weighted	120.00
HC1_weighted	283.00
HC13_weighted	16.00
HC2_weighted	25.00
HC3_weighted	1.00
SP1_weighted	6618.00
HC15_weighted	0.00



Feature	Value
SP2_weighted	2.00
SP1_weighted	7.00
HC3_weighted	0.00
HC1_weighted	219.00
HB3_weighted	0.00
HC4_weighted	0.00
HC13_weighted	6.00
HC2_weighted	1.00

11 K-means clustering

Our second model is a K-means clustering model.

Similarly to the previous model, we start by making a copy of the dataframe

```
df = df_joined.copy()
```

Pull the date from the DF

```
df['day_of_week'] = df['date'].dt.dayofweek

df['month'] = df['date'].dt.month

df['year'] = df['date'].dt.year

df.drop(columns=['date'], inplace=True)
```

Cluster based on latitude and longitude.

```
df_clustering = df[['latitude', 'longitude']]
```

Cluster 5 different groups together with the model and plot this on a scatterplot, because this is based on latitude and longitude it resembles the way it would look like on a map.

```
n_clusters = 5  # Adjust the number of clusters as needed

kmeans = KMeans(n_clusters=n_clusters, random_state=42)

df['cluster'] = kmeans.fit_predict(df_clustering_scaled)


plt.figure(figsize=(12, 8))

plt.scatter(df['latitude'], df['longitude'], c=df['cluster'],
cmap='viridis', marker='o', edgecolor='k')

plt.xlabel('Latitude')

plt.ylabel('Longitude')

plt.title('Clustering of Incidents')

plt.colorbar(label='Cluster Label')

plt.show()
```

Then we have a hierarchical clustering. Once again we take the same dates as before, and cluster latitude and longitude. We standardise he values with

```
scaler = StandardScaler()
df_clustering_scaled=scaler.fit_transform(df_clustering)
```

Check the model based on 20000 samples and parse through those 20000.

```
sample_size = 20000  # Adjust based on your memory capacity

if len(df_clustering_scaled) > sample_size:

    sample_indices = np.random.choice(df_clustering_scaled.shape[0], 
sample_size, replace=False)

    df_clustering_sampled = df_clustering_scaled[sample_indices]

else:

    df_clustering_sampled = df_clustering_scaled
```

Then we use the ward method for clustering, this minimises variance within clusters, then the dendogram is plotted. Dendograms plot based on hierarchy, which allows one way of seeing the clusters. Next we establish a scatterplot on a map like we did before.

12 Decision tree model

Our third model is a decision tree model. The model uses road_name, risk_category_encoded, weather_code and risk_level_wmo, to calculate incidents based on road_name. This is done by train_test_split. Then after that a bar chart is plotted to show the distribution of classes. Class 1 has significantly fewer classes than other classes so SMOTE is applied to equalise the distribution. Then we set up the random forest for predicting y_test for the dataset.

```
y_pred_test_rf = rf.predict(X_test)
```

The model then shows accuracy score, as well as full classification with precision, recall and f1-score. We then plot a confusion matrix for the risk level classification to gain insight on how things are plotted, and have a feature importance chart.

13 DNN

After the basic models, we implemented a deep neural network.

In the first part of the code we select the columns that will be used for the DNN.

Then selected categorical variables are converted to numerical variables.

```
encoder = LabelEncoder()

df_selected['incident_severity'] =
encoder.fit_transform(df_selected['incident_severity'])

df_selected['road_name'] =
encoder.fit_transform(df_selected['road_name'].astype(str))

df_selected['risk_level_wmo'] =
encoder.fit_transform(df_selected['risk_level_wmo'])
```

After that we extract the date information by having days, months and years separately from each other. After this we drop the risk_category_encoded column for X as it will be used for the Y value. It is then one-hot encoded as it is more suitable for classification with multiple classes.

```
X = df_selected.drop(columns=['risk_category_encoded'])  # Features

y = df_selected['risk_category_encoded']  # Target

# One-hot encode the target variable

y_encoded = to_categorical(y)
```

Then we use train_test_split with an 80% training and 20% testing split, standardise the features, and build the model.

The input layer is 128 neurons with ReLU activation, then a dropout layer of 0.5 to prevent overfitting, then a 64 neuron layer, another dropout layer, and another layer with 32 neurons, before reaching the output layer with softmax activation.

The model is then compiled using categorical crossentropy, adam optimiser and accuracy as metric so we can compare it with the other models we made. We print the loss and accuracy and a heatmap.

Iteration 2

After this first iteration we made an L2 version of the model, L2 penalises large weights which prevents overfitting. Early stopping is introduced to reduce computing time, as well as reducing overfitting. Both changes lead to an improvement in the output of the model.

14 RNN

Lastly, we made an RNN

For the RNN we converted the times to the date format, sorted our values by the date, and put a label encoder in place to convert categorical value of incident severity into numeric values.

```
label_encoder = LabelEncoder()

df_joined['incident_severity'] =
label_encoder.fit_transform(df_joined['incident_severity'])
```

After this we selected all the features that would be relevant for analysis, used MinMaxScaler() for normalisation. After this we created a sequence for time series analysis.

Data is our input data
Target is the target values based on the data

```
features = ['latitude', 'longitude', 'duration_seconds', 'maxwaarde',
'temperature_2m',
    'rain', 'snowfall', 'snow_depth', 'risk_category_encoded',
'risk_level_wmo']
  target = 'incident_severity'
```

Sequence_length is 10 as a default.

Sequences is a list storing the sequences of features

Targets is a list which sorts target values at the end of each sequence.

This then iterates over the data and creates sequences of length 10.

After that we generate sequences and the targets from the data to use in train_test_split.

```
model = Sequential()
      model.add(LSTM(50, return_sequences=True,
      input_shape=(sequence_length,
      len(features))))
        model.add(Dropout(0.2))
        model.add(LSTM(50, return_sequences=False))
        model.add(Dropout(0.2))
        model.add(Dense(1, activation='linear'))

        model.compile(optimizer='adam', loss='mse')
        model.summary()
```

The next code builds the model with the first LSTM layer having 50 units, outputs are sequenced to the next LSTM layer. The dropout drops 20% of neurons randomly to prevent overfitting, then there is the second LSTM layer which does not sequence outputs to he next layer as there is no 3rd layer. Then another equivalent dropout layer before going to the output with a dense of 1, as the RNN predicts continuous values.
The model is then compiled using adam optimser and MSE loss, and a summary is printed. The model is then trained using 20 epochs, and then evaluated based on the loss, and predictions are made and plotted on the table.

Then future incident severity is predicted, and transformed back to the original scale for interpretability.

Iteration 2

In the second iteration of the RNN has some fewer features to focus on those and see if that makes a difference. Event_id, road_name, risk_level_wmo have been cut. Then we establish model architecture and train like we did in the previous one, then we print actual and predicted values, with one actual value being much higher than the predicted value. We then look at mae, mse, rmse and mape and print those values and plot the distribution of residuals.

Iteration 3

In the third iteration the target variable is risk_level_wmo instead of incident_severity. We switched to using MinMaxScaler instead of StandardScaler, we also use forward filling in case a value shows up as missing somehow.

15 Model Comparison

After comparing the best performances of all three models in terms of evaluation metrics, we can conclude that **the Random Forest (3rd iteration)** can be considered the best-performing one, in terms of evaluation results and complexity.

The **Decision Trees model (3rd iteration)**, tuned using RandomizedSearchCV, caps its overall **test accuracy of 97.38%.** The high precision, recall and F1-score across all classes indicate that the model is effective at distinguishing classes.

The **DNN model (2nd iteration)** shows impressive results with **98.86% accuracy** and overall high metrics, especially for class 0 (low risk) and 2 (high risk). However, due to its added complexity as a neural network model with L2 regularization, it is not better feasibility-wise than a standard machine learning model.

Therefore, the **Random Forest model (3rd iteration)**, with the highest accuracy out of all basic ML models **(98.8%),** exhibits robust precision, recall, and F1-score across all classes. Furthermore, the cross-validation scores are high, with an **average score of 98.94%** across all five folds. The complexity of the model is also appropriate and can be more easily moulded on multiple datasets. When discussing transparency, the model's inner workings are further explained in the XAI/Interpretability section. This model provides high accuracy, robust performance and simple Interpretability for the developer and the user.

16 Sources

EU Artificial Intelligence Act | Up-to-date developments and analyses of the EU AI Act. (n.d.). https://artificialintelligenceact.eu/