Deployment & Interface Design

Deployment & Interface Design - 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 the deployment and interface design of models used within an AI system developed to improve road safety in Breda. As we provide a detailed description of the deployment process and the interface design strategies employed to enhance the user experience and overall system effectiveness.

2 Goal

The goal of this file is to document the steps taken in the deployment and interface design phases of the project to improve road safety in Breda. This is to fulfil regulatory requirements and to ensure user-friendly usage, providing assurance that the system will effectively enhance their driving safety.

3 Method

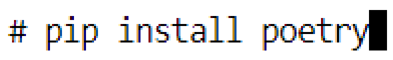
To ensure effective deployment and interface design for the road safety AI system in Breda, the process begins with defining hypotheses to determine what to test and the expected outcomes. The experiment design involves randomly assigning users to either Version A or B of the interface. Data collection follows, gathering feedback on a scale from 1 to 7, which is then analysed by calculating means, standard deviations, and performing t-tests. Based on these results, the interface is adjusted through iterative improvements to continuously refine the design. A demo application prototype is developed and tested to finalize the interface.

For deployment, the first step is creating a virtual environment to ensure isolation and consistency, followed by installing the necessary dependencies. Code cleaning involves refactoring to optimize performance and readability. Unit testing is conducted by developing and automating test cases to ensure all functionalities work as expected. The final deployment includes deploying the application to the cloud for scalability and availability, setting up monitoring tools to track performance, and providing user training along with comprehensive documentation to facilitate effective system use.

4 Deployment - Environment Creation

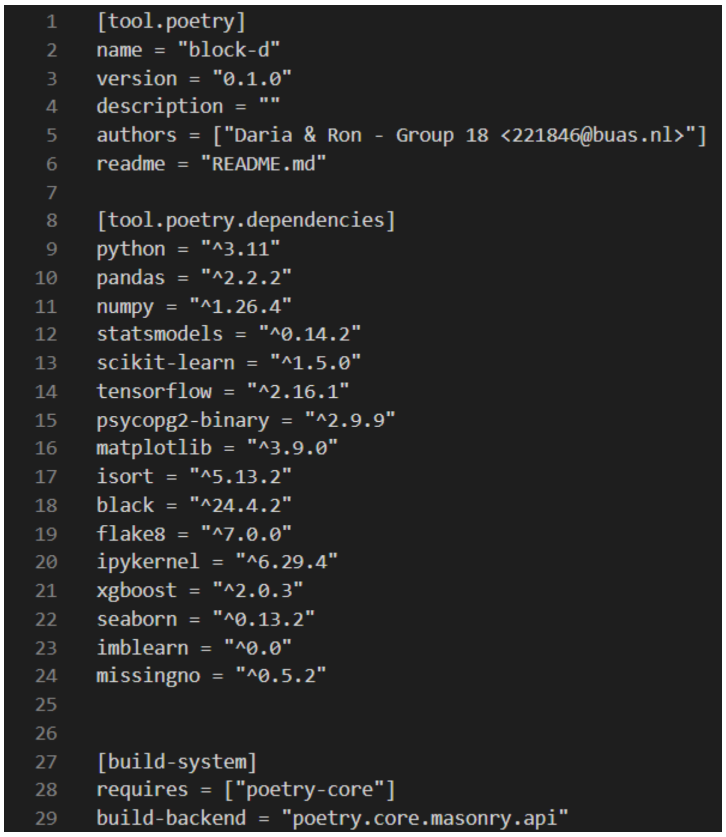
Creating an isolated virtual environment is a critical step in the deployment process of the road safety AI system in Breda. This environment ensures that the deployment process is consistent and does not interfere with existing systems. A virtual environment provides a dedicated space where all dependencies and libraries specific to the project can be installed and managed independently of other projects. This isolation helps in avoiding conflicts between different software versions and configurations that may exist on the host system.

First step taken was to install poetry in our virtual server using the terminal:



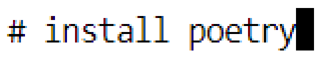
```
# pip install poetry
```

when package installation is complete, we write the TOML file that specify which additional packages are need for the environment:



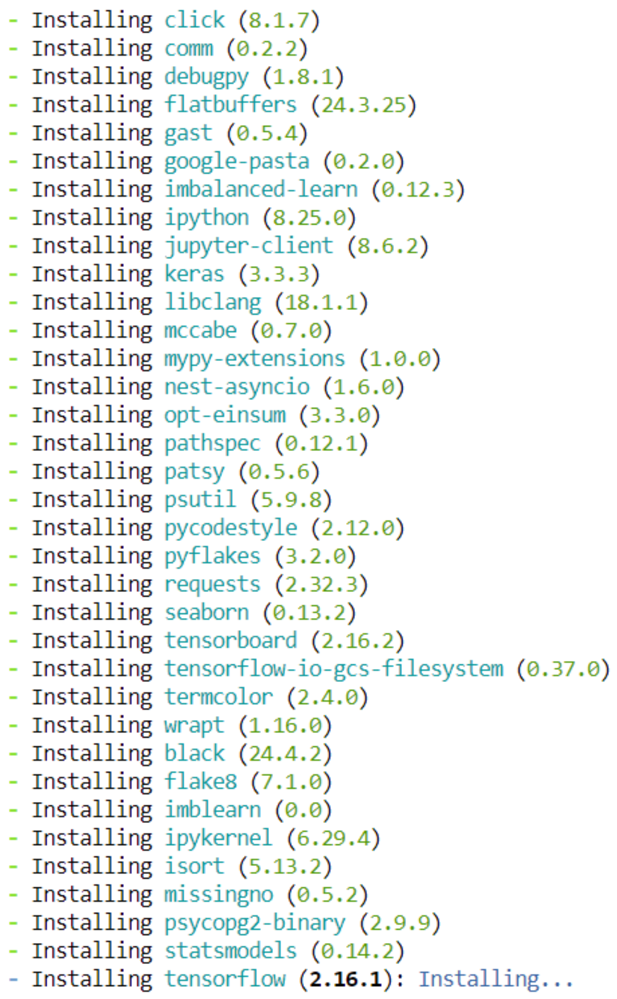
```
[tool.poetry]
name = "block-d"
version = "0.1.0"
description = ""
authors = ["Daria & Ron - Group 18 <221846@buas.nl>"]
readme = "README.md"

[tool.poetry.dependencies]
python = "^3.11"
pandas = "^2.2.2"
numpy = "^1.26.4"
statsmodels = "^0.14.2"
scikit-learn = "^1.5.0"
tensorflow = "^2.16.1"
psycopg2-binary = "^2.9.9"
matplotlib = "^3.9.0"
isort = "^5.13.2"
black = "^24.4.2"
flake8 = "^7.0.0"
ipykernel = "^6.29.4"
xgboost = "^2.0.3"
seaborn = "^0.13.2"
imblearn = "^0.0"
missingno = "^0.5.2"


[build-system]
requires = ["poetry-core"]
build-backend = "poetry.core.masonry.api"
```

When running the command “# install poetry” in the right directory, poetry will locate the TOML file, read it and run installation for the mentioned packages:



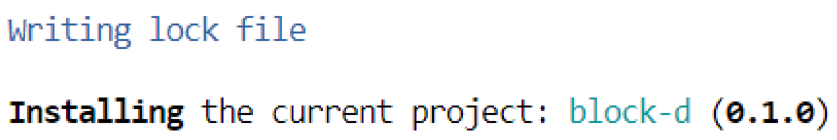
```
# install poetry
```

Installation of the packages will begin, and the terminal will provide the status of each package:



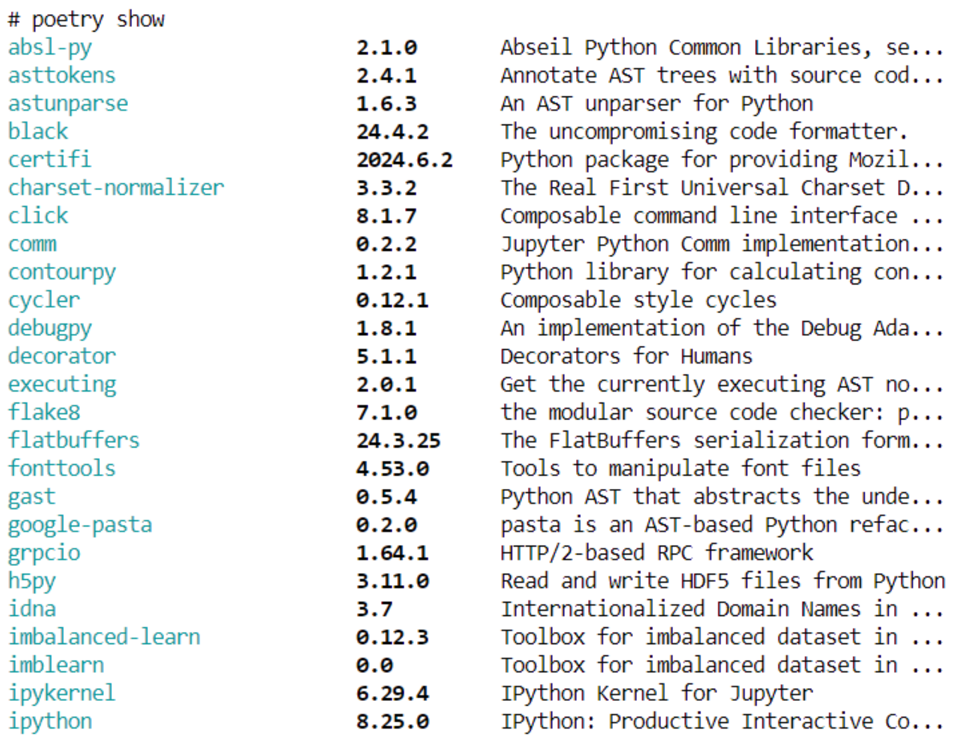
```
- Installing click (8.1.7)
- Installing comm (0.2.2)
- Installing debugpy (1.8.1)
- Installing flatbuffers (24.3.25)
- Installing gast (0.5.4)
- Installing google-pasta (0.2.0)
- Installing imbalanced-learn (0.12.3)
- Installing ipython (8.25.0)
- Installing jupyter-client (8.6.2)
- Installing keras (3.3.3)
- Installing libclang (18.1.1)
- Installing mccabe (0.7.0)
- Installing mypy-extensions (1.0.0)
- Installing nest-asyncio (1.6.0)
- Installing opt-einsum (3.3.0)
- Installing pathspec (0.12.1)
- Installing patsy (0.5.6)
- Installing psutil (5.9.8)
- Installing pycodestyle (2.12.0)
- Installing pyflakes (3.2.0)
- Installing requests (2.32.3)
- Installing seaborn (0.13.2)
- Installing tensorboard (2.16.2)
- Installing tensorflow-io-gcs-filesystem (0.37.0)
- Installing termcolor (2.4.0)
- Installing wrapt (1.16.0)
- Installing black (24.4.2)
- Installing flake8 (7.1.0)
- Installing imblearn (0.0)
- Installing ipykernel (6.29.4)
- Installing isort (5.13.2)
- Installing missingno (0.5.2)
- Installing psycopg2-binary (2.9.9)
- Installing statsmodels (0.14.2)
- Installing tensorflow (2.16.1): Installing...
```

Poetry.lock is created or updated during this process. This file locks the exact versions of all dependencies (including sub-dependencies) used in your project:



```
Writing lock file

Installing the current project: block-d (0.1.0)
```

To confirm the last package installation, we use the command “# poetry show” in the terminal, as it provides a list of the current packages in the environment.

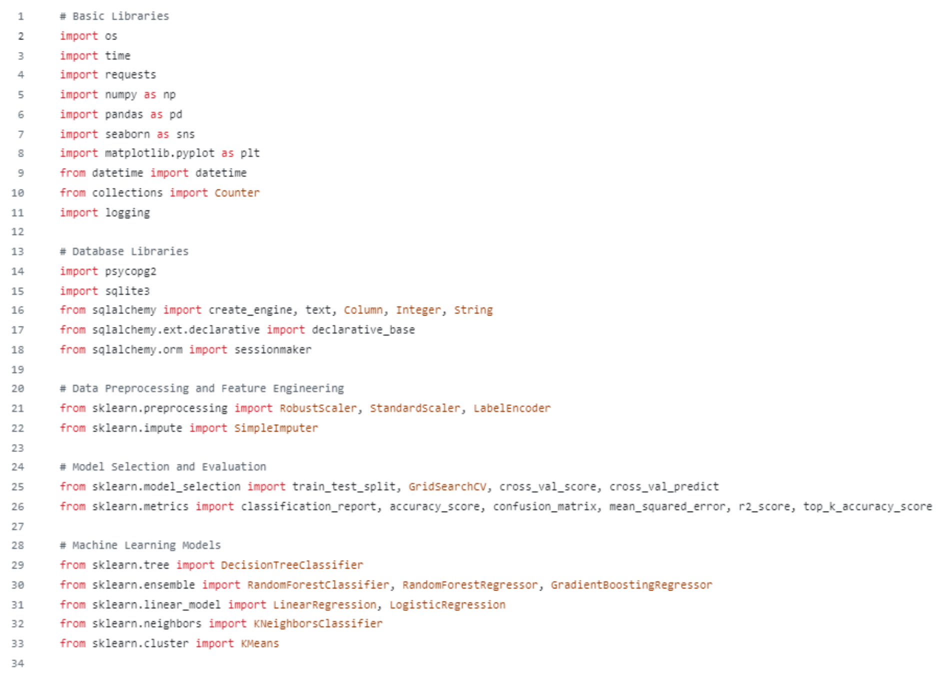


```
# poetry show
absl-py                      2.1.0       Abseil Python Common Libraries, se...
asttokens                    2.4.1       Annotate AST trees with source cod...
astunparse                   1.6.3       An AST unparser for Python
black                        24.4.2      The uncompromising code formatter.
certifi                      2024.6.2    Python package for providing Mozil...
charset-normalizer           3.3.2       The Real First Universal Charset D...
click                        8.1.7       Composable command line interface ...
comm                         0.2.2       Jupyter Python Comm implementation...
contourpy                    1.2.1       Python library for calculating con...
cycler                       0.12.1      Composable style cycles
debugpy                      1.8.1       An implementation of the Debug Ada...
decorator                    5.1.1       Decorators for Humans
executing                    2.0.1       Get the currently executing AST no...
flake8                       7.1.0       the modular source code checker: p...
flatbuffers                  24.3.25     The FlatBuffers serialization form...
fonttools                    4.53.0      Tools to manipulate font files
gast                         0.5.4       Python AST that abstracts the unde...
google-pasta                 0.2.0       pasta is an AST-based Python refac...
grpcio                       1.64.1      HTTP/2-based RPC framework
h5py                         3.11.0      Read and write HDF5 files from Python
idna                         3.7         Internationalized Domain Names in ...
imbalanced-learn             0.12.3      Toolbox for imbalanced dataset in ...
imblearn                     0.0         Toolbox for imbalanced dataset in ...
ipykernel                    6.29.4      IPython Kernel for Jupyter
ipython                      8.25.0      IPython: Productive Interactive Co...
```

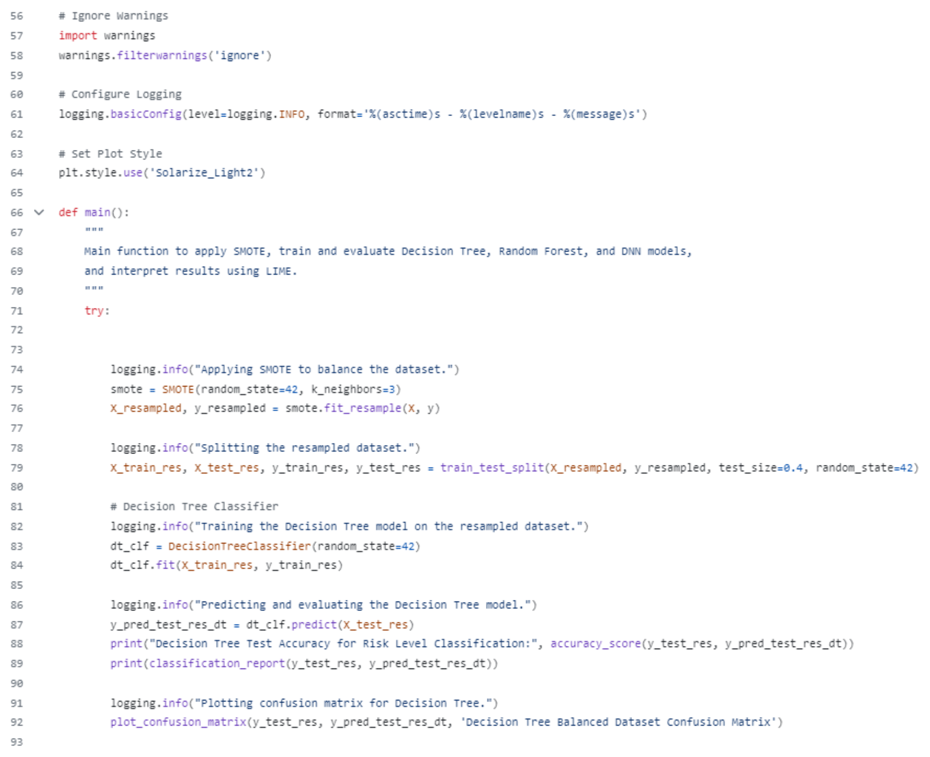
For last, we save the environment and assign a name to it by writing the following command in the server’s terminal # poetry run python -m ipykernel install --user --name PoetryEnvironment --display-name "Poetry Environment"

5 Code Cleaning

Code cleaning is the process of restructuring existing computer code without changing its external behaviour. This involves making the code more readable, maintainable, and efficient. Good code cleaning practices help developers understand the code better, reduce complexity, and eliminate redundant or obsolete code, which ultimately leads to fewer bugs and easier maintenance. Example for used code cleaning for our project:



```python
# Basic Libraries
import os
import time
import requests
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import datetime
from collections import Counter
import logging

# Database Libraries
import psycopg2
import sqlite3
from sqlalchemy import create_engine, text, Column, Integer, String
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

# Data Preprocessing and Feature Engineering
from sklearn.preprocessing import RobustScaler, StandardScaler, LabelEncoder
from sklearn.impute import SimpleImputer

# Model Selection and Evaluation
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score, cross_val_predict
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, mean_squared_error, r2_score, top_k_accuracy_score

# Machine Learning Models
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor, GradientBoostingRegressor
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans

```



```python
# Ignore Warnings
import warnings
warnings.filterwarnings('ignore')

# Configure Logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

# Set Plot Style
plt.style.use('Solarize_Light2')

def main():
    """
    Main function to apply SMOTE, train and evaluate Decision Tree, Random Forest, and DNN models,
    and interpret results using LIME.
    """
    try:


        logging.info("Applying SMOTE to balance the dataset.")
        smote = SMOTE(random_state=42, k_neighbors=3)
        X_resampled, y_resampled = smote.fit_resample(X, y)

        logging.info("Splitting the resampled dataset.")
        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)

        # Decision Tree Classifier
        logging.info("Training the Decision Tree model on the resampled dataset.")
        dt_clf = DecisionTreeClassifier(random_state=42)
        dt_clf.fit(X_train_res, y_train_res)

        logging.info("Predicting and evaluating the Decision Tree model.")
        y_pred_test_res_dt = dt_clf.predict(X_test_res)
        print("Decision Tree Test Accuracy for Risk Level Classification:", accuracy_score(y_test_res, y_pred_test_res_dt))
        print(classification_report(y_test_res, y_pred_test_res_dt))

        logging.info("Plotting confusion matrix for Decision Tree.")
        plot_confusion_matrix(y_test_res, y_pred_test_res_dt, 'Decision Tree Balanced Dataset Confusion Matrix')

```

6 Unit Testing

Unit testing is a software testing method where individual units or components of the software are tested in isolation to ensure they function correctly. The main goal is to validate that each unit of the software performs as expected and to identify any issues early in the development process.

Setup the Testing Environment:

Installed necessary testing libraries, including unittest for writing and running tests and coverage for measuring code coverage.

Created Test Cases:

Developed test cases in test_main_script.py to validate the functionality of different components of the main_script.py module.

Ensured each test case targets a specific function or feature, such as data loading, data preprocessing, model training, and evaluation.

Running the Tests:

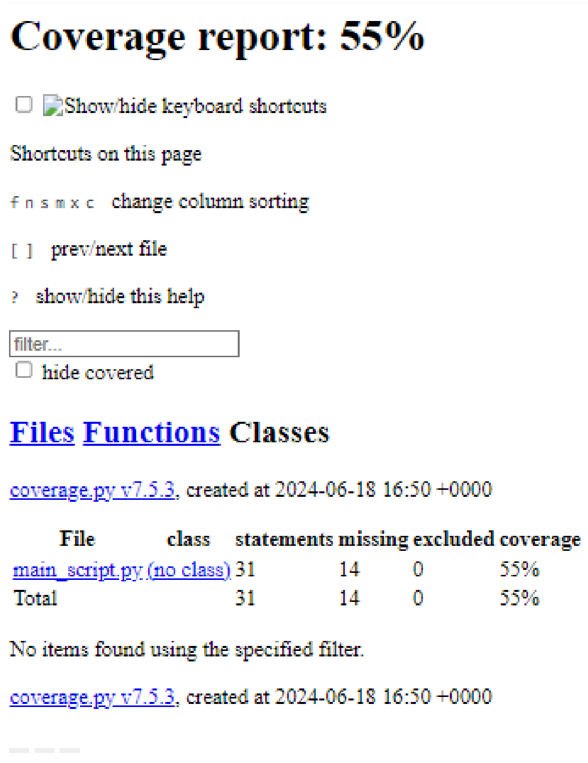
Executed the test suite using unittest to verify that all test cases pass, and the software behaves as expected.

Utilized the coverage tool to run the tests and collect coverage data, ensuring that all parts of the code are tested.

Generating and Viewing Coverage Reports:

GeneratedwasHTML coverage report using coverage html to visually inspect which parts of the code were covered by the tests.

Opened the generated htmlcov/index.html file in a web browser to review the detailed coverage report, identifying areas with insufficient test coverage.



Coverage report: 55%

☐ Show/hide keyboard shortcuts

Shortcuts on this page

`f n s m x c` change column sorting

`[ ]` prev/next file

`?` show/hide this help

filter...

☐ hide covered

Files Functions Classes

coverage.py v7.5.3, created at 2024-06-18 16:50 +0000

File	class	statements	missing	excluded	coverage
main_script.py	(no class)	31	14	0	55%
Total		31	14	0	55%

No items found using the specified filter.

coverage.py v7.5.3, created at 2024-06-18 16:50 +0000

By implementing unit testing and achieving 55% code coverage, we ensured that our AI system for road safety in Breda is robust, reliable, and maintainable, providing assurance that the model will help users drive more safely.

7 Interface Design

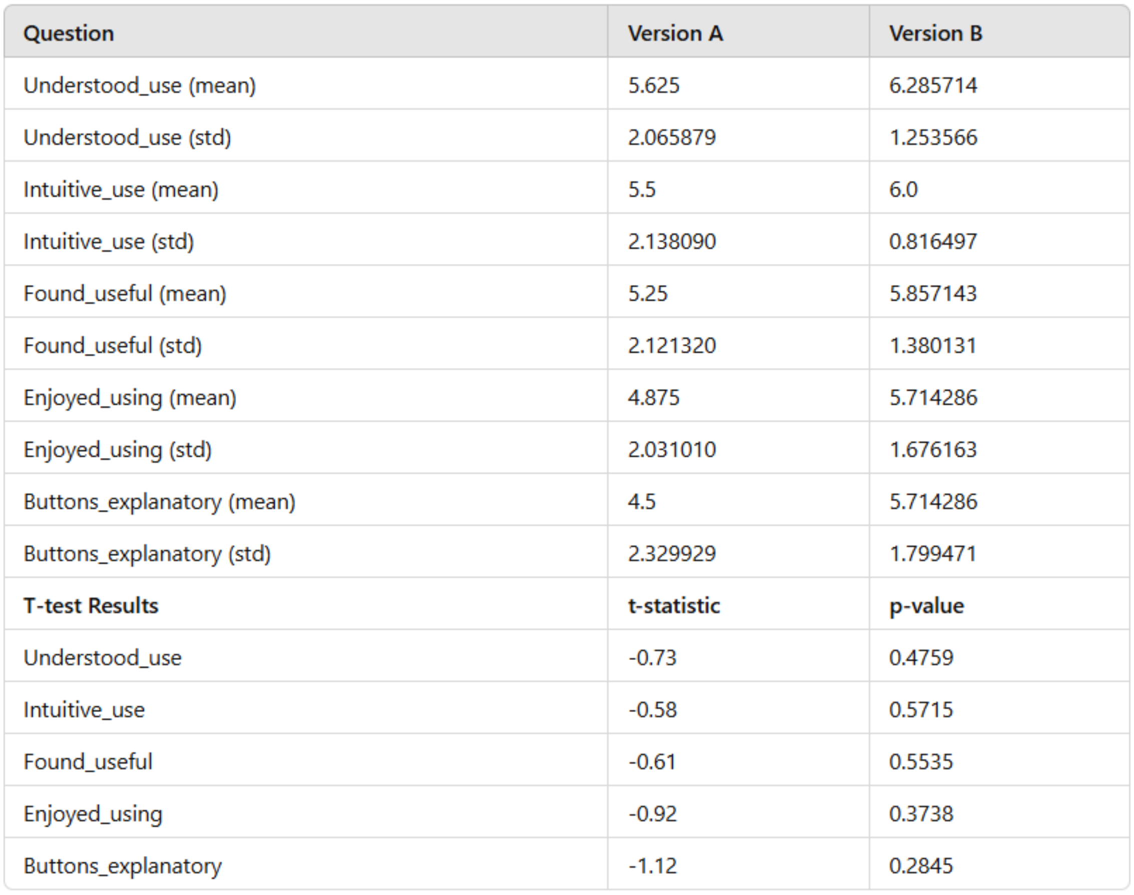
Data Analysis:

We used the following steps for data analysis:

- **Combine Data**: Integrate data from both versions into a single dataset.
- **Calculate Descriptive Statistics**: Compute means and standard deviations for each question for both versions.
- **Perform T-tests**: Conduct t-tests to compare the means of the two groups for each question.

Link to AB-test analysis

Results:



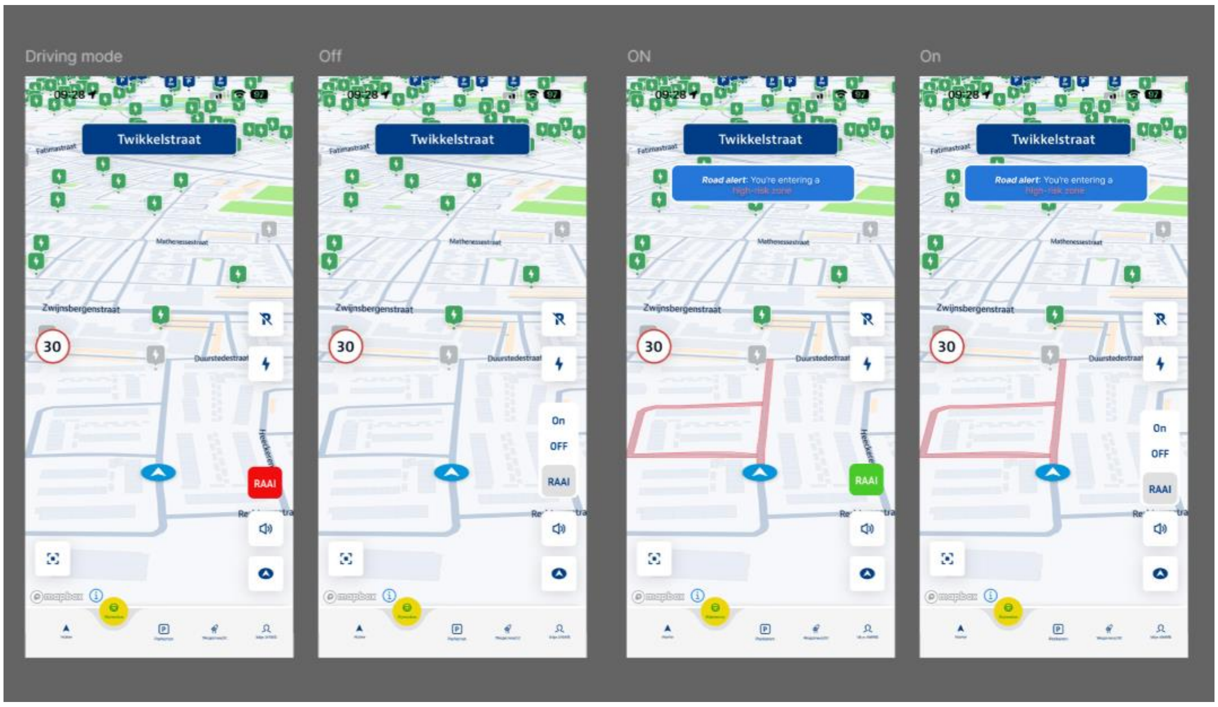
Question	Version A	Version B
Understood_use (mean)	5.625	6.285714
Understood_use (std)	2.065879	1.253566
Intuitive_use (mean)	5.5	6.0
Intuitive_use (std)	2.138090	0.816497
Found_useful (mean)	5.25	5.857143
Found_useful (std)	2.121320	1.380131
Enjoyed_using (mean)	4.875	5.714286
Enjoyed_using (std)	2.031010	1.676163
Buttons_explanatory (mean)	4.5	5.714286
Buttons_explanatory (std)	2.329929	1.799471
T-test Results	**t-statistic**	**p-value**
Understood_use	-0.73	0.4759
Intuitive_use	-0.58	0.5715
Found_useful	-0.61	0.5535
Enjoyed_using	-0.92	0.3738
Buttons_explanatory	-1.12	0.2845

Adjustments:

Based on the findings from the A/B test, several adjustments were made to improve the app's user experience.

Addition of RAAI Button

- **Feature**: A Road Assistance using Artificial Intelligence (RAAI) button was added in the driving mode.
- **Functionality**: Users can turn the RAAI feature on or off by pressing this button.
- **Purpose**: The RAAI feature provides enhanced road assistance, aiming to improve safety and user convenience.
- **User Feedback**: Preliminary feedback from users who tested the RAAI feature has been positive, indicating that it adds significant value to the driving mode experience.



Conclusion:

The results indicate that Version B tends to have higher mean scores across all questions compared to Version A, despite the p-values indicating that the differences are not statistically significant at a conventional level. However, the consistent higher scores for Version B suggest a preference for Version B among users.

Based on the analysis of the data, Version B is preferred over Version A due to higher average ratings in all measured aspects. While the statistical significance is not strong, the trend in the data indicates that Version B provides a better user experience overall.