Table of Contents

1. Lab 1 - Explainable AI

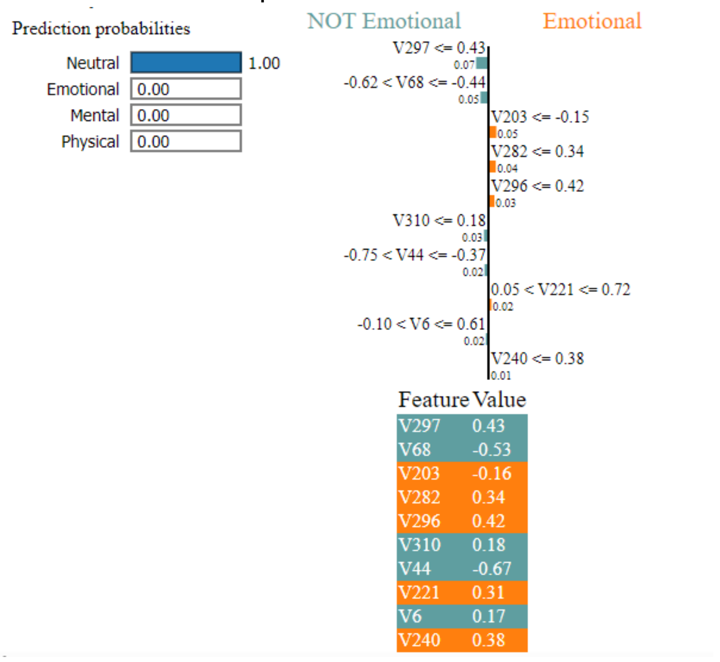
Introduction

In this section of report, we compare two machine learning models, Decision Tree and Random Forest, for activity recognition based on wearable physio data. The analysis focuses on explainability using different XAI methods like LIME and feature importance techniques. We will evaluate how explanations differ across the models, both globally and locally, and assess the models robustness to noisy data. We will also explore misclassifications to understand potential differences between the models predictions and explanations.

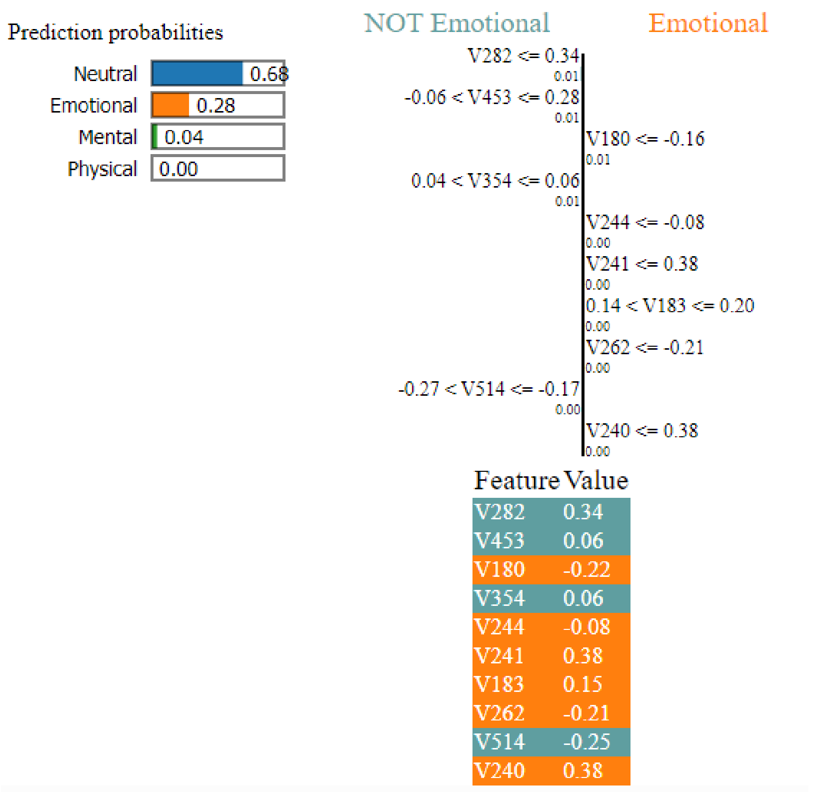
Do Explanations Look Reasonable?

We analyse the local feature explanations generated by LIME for both models. For Random Forest, features like V282, V297, and V240 show an influence on predicting emotional and mental states. These features affect the predicted probabilities if you see the LIME plots below. Also, for Decision Tree, V68, V259, and V203 have important roles in explaining individual predictions. The explanations for LIME seem reasonable as they align well with the feature distributions and the models' behaviour.
Plots:

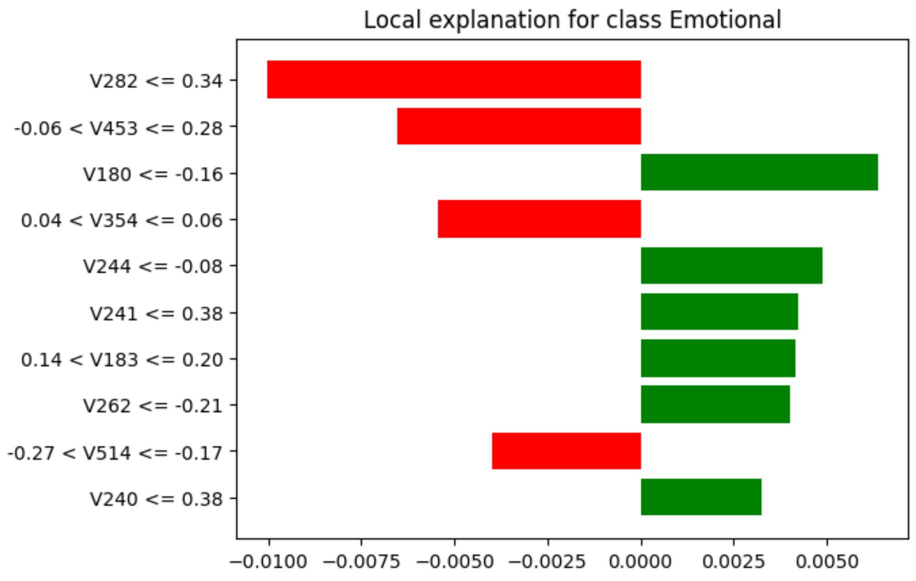
- Local LIME explanations for Random Forest and Decision Tree:

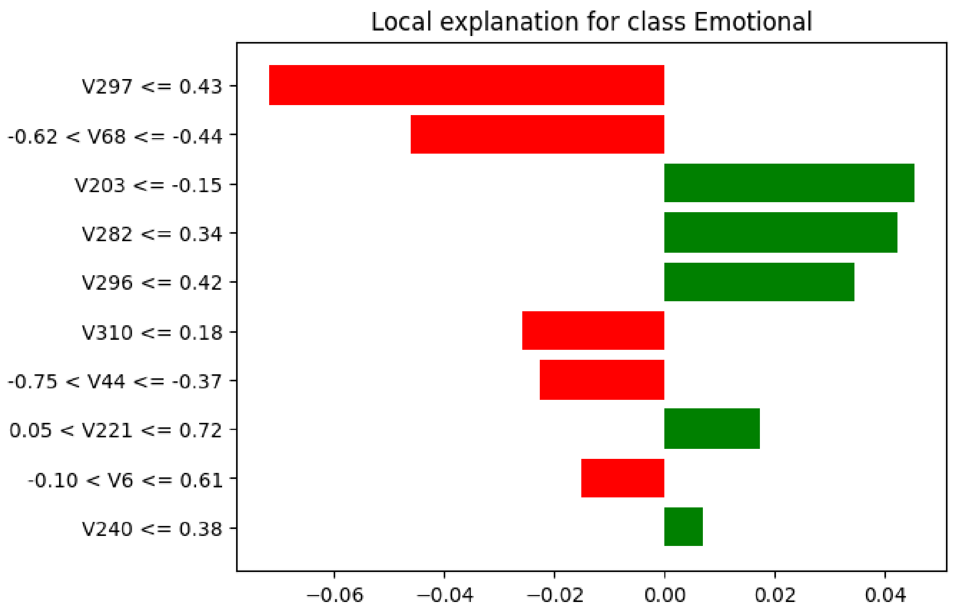


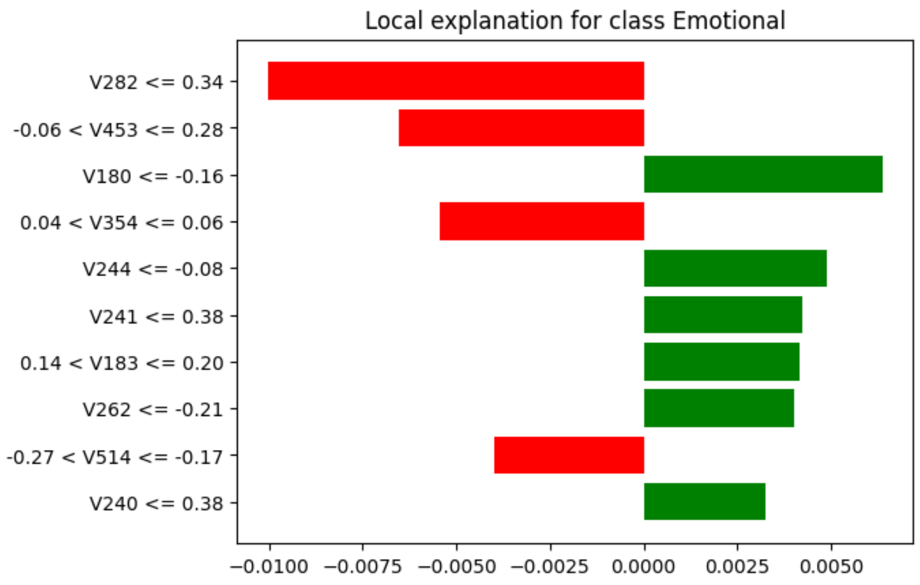
Feature	Value
V297	0.43
V68	-0.53
V203	-0.16
V282	0.34
V296	0.42
V310	0.18
V44	-0.67
V221	0.31
V6	0.17
V240	0.38



Feature	Value
V282	0.34
V453	0.06
V180	-0.22
V354	0.06
V244	-0.08
V241	0.38
V183	0.15
V262	-0.21
V514	-0.25
V240	0.38



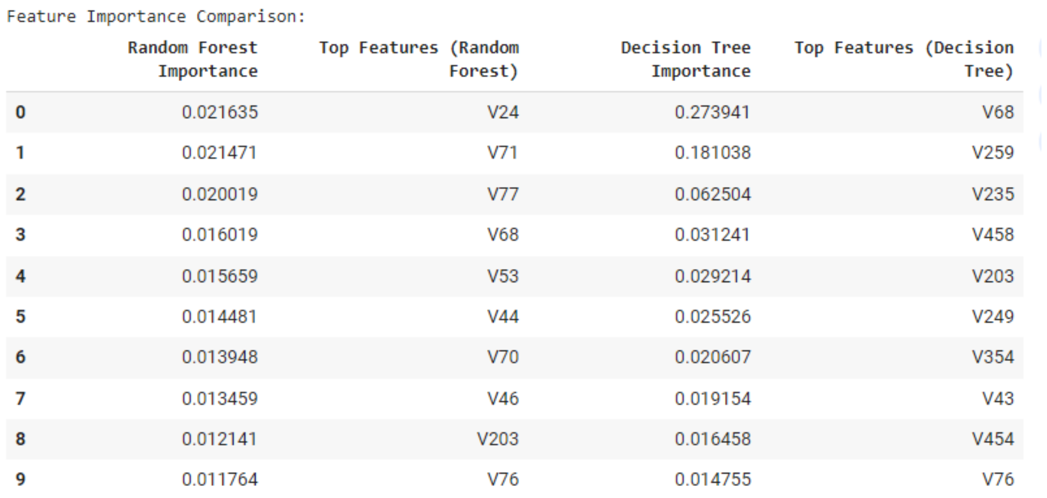




Are Explanations Stable Across XAI Methods?

Now we will compare the explanations generated by LIME with global feature importance methods Random Forest and Decision Tree feature importance. For Random Forest, V24, V71, and V77 are globally important, whereas LIME emphasizes V241, V297, and V282. This suggests some stability between methods, as V282 and V297 appear in both LIME and global importance rankings. For Decision Tree, V68 and V259 dominate both LIME and global importance rankings, showing stability between methods for this model.
Plots:

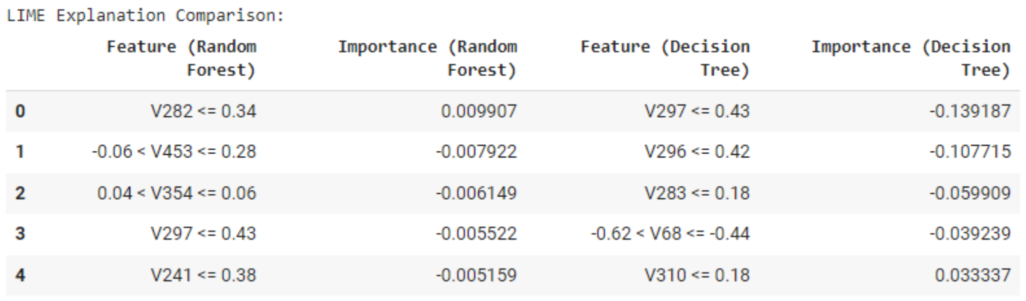
- Feature Importance Comparison:



Feature Importance Comparison:

	Random Forest Importance	Top Features (Random Forest)	Decision Tree Importance	Top Features (Decision Tree)
0	0.021635	V24	0.273941	V68
1	0.021471	V71	0.181038	V259
2	0.020019	V77	0.062504	V235
3	0.016019	V68	0.031241	V458
4	0.015659	V53	0.029214	V203
5	0.014481	V44	0.025526	V249
6	0.013948	V70	0.020607	V354
7	0.013459	V46	0.019154	V43
8	0.012141	V203	0.016458	V454
9	0.011764	V76	0.014755	V76

- LIME Explanation Comparison:



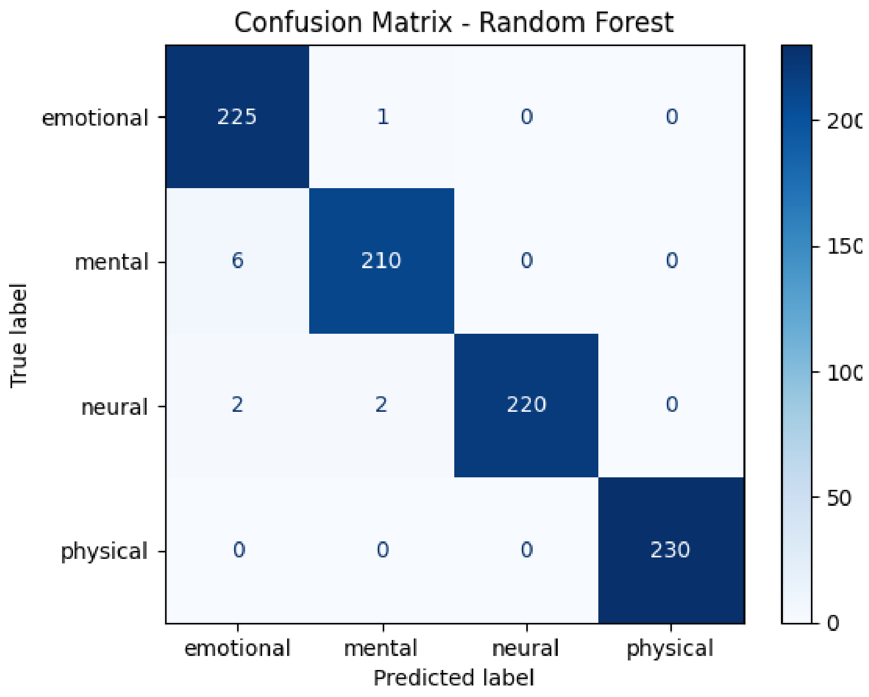
LIME Explanation Comparison:

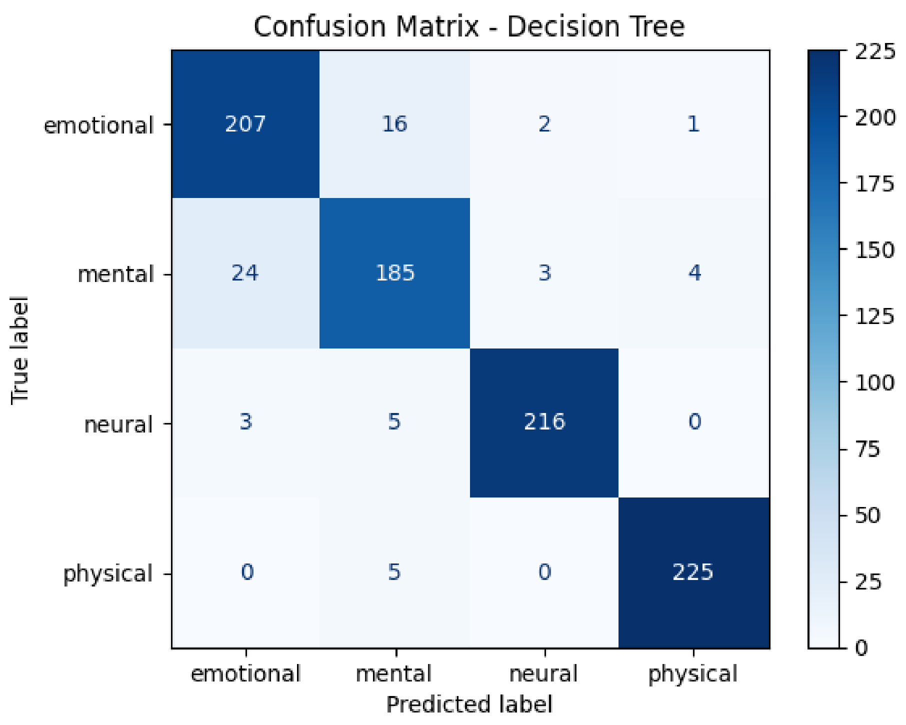
	Feature (Random Forest)	Importance (Random Forest)	Feature (Decision Tree)	Importance (Decision Tree)
0	V282 <= 0.34	0.009907	V297 <= 0.43	-0.139187
1	-0.06 < V453 <= 0.28	-0.007922	V296 <= 0.42	-0.107715
2	0.04 < V354 <= 0.06	-0.006149	V283 <= 0.18	-0.059909
3	V297 <= 0.43	-0.005522	-0.62 < V68 <= -0.44	-0.039239
4	V241 <= 0.38	-0.005159	V310 <= 0.18	0.033337

Are Explanations Stable Across Machine Learning Methods?

Plots:

- LIME Explanation Comparison Table:

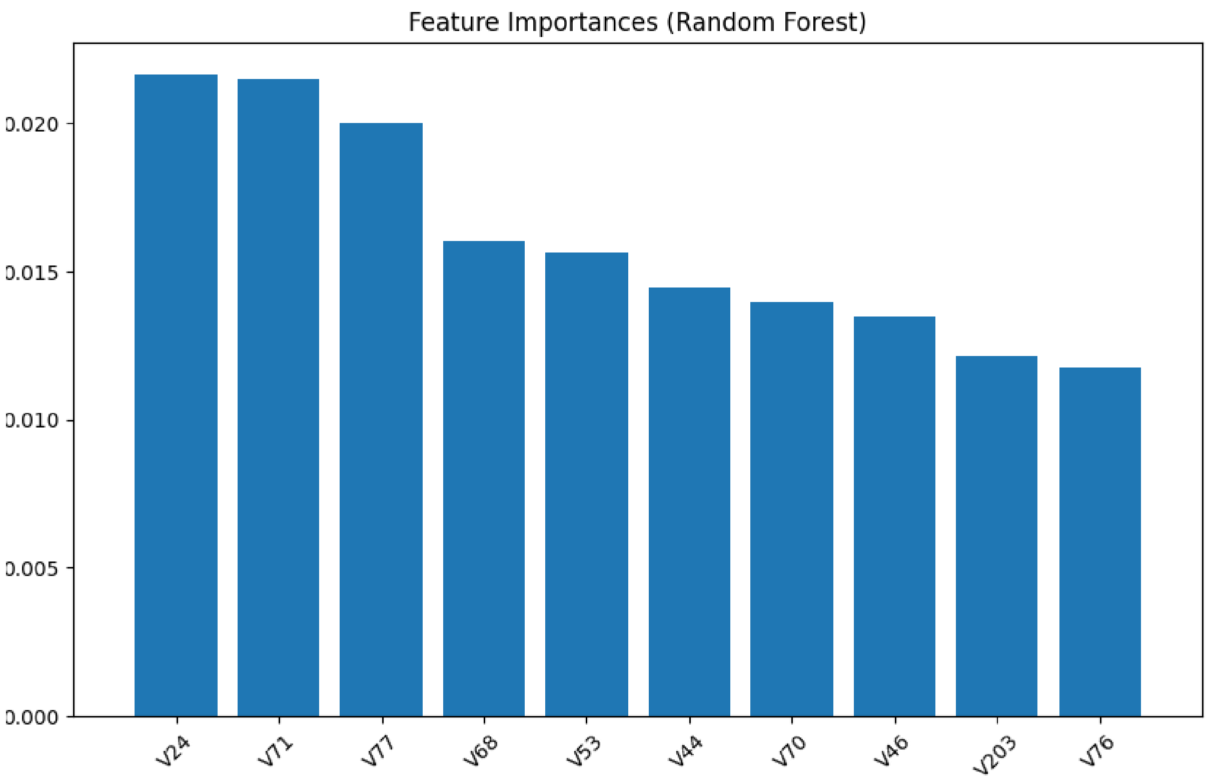


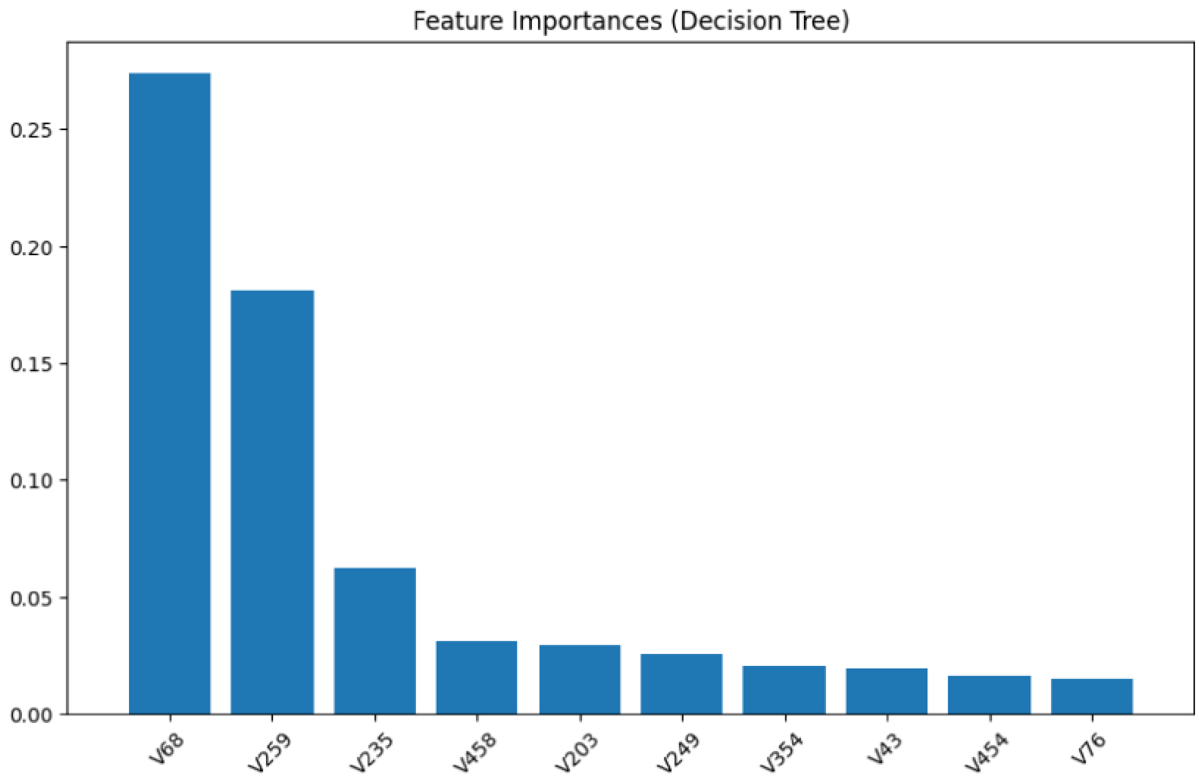


What Are the Most Important/Informative Features, and What Is Their Influence on the Results: Global vs. Local Individual Explanations?

The most informative features globally and locally reveal how different features contribute to overall model behaviour and specific predictions. For Random Forest, globally important features like V24, V71, and V77 contribute to the model's overall performance. In Decision Tree, V68 and V259 are both globally and locally important. This highlights that the global importance of a feature doesn't always reflect its influence on individual decisions.
Plots:

- Feature Importance Comparison:

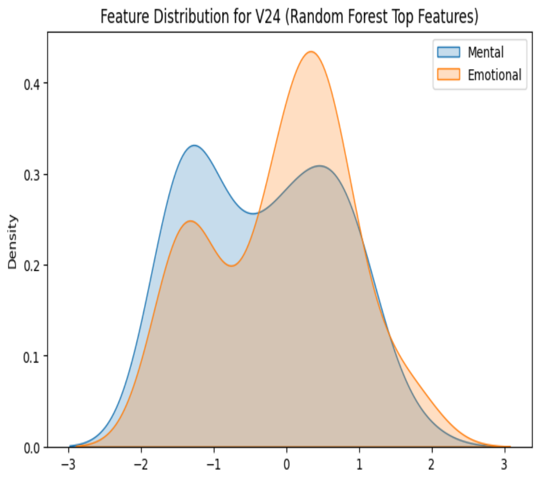

Feature Importances (Random Forest)
0.000
0.005
0.010
0.015
0.020
V24
V71
V77
V68
V53
V44
V70
V46
V203
V76

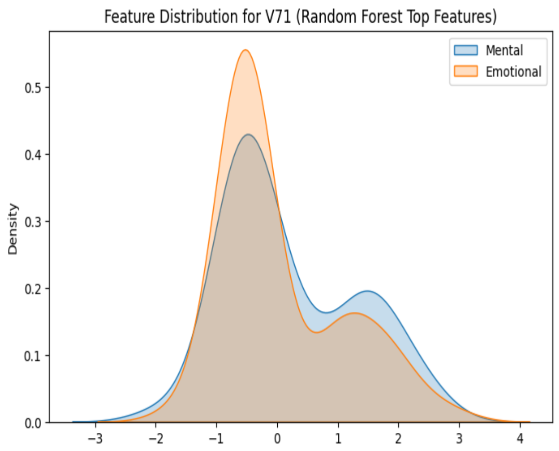

Feature Importances (Decision Tree)
0.00
0.05
0.10
0.15
0.20
0.25
V68
V259
V235
V458
V203
V249
V354
V43
V454
V76

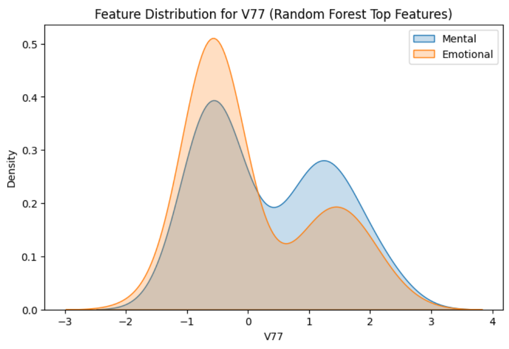
Feature Importance Comparison:

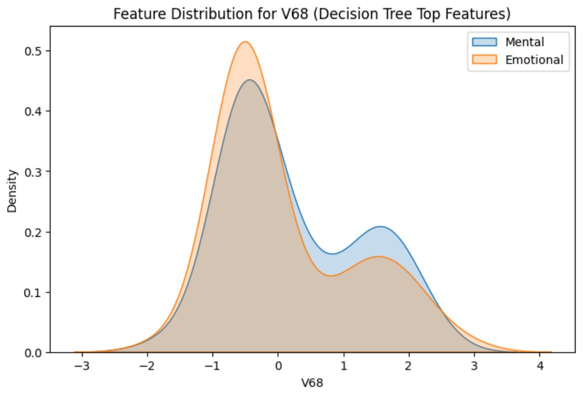
	Random Forest Importance	Top Features (Random Forest)	Decision Tree Importance	Top Features (Decision Tree)
0	0.021635	V24	0.273941	V68
1	0.021471	V71	0.181038	V259
2	0.020019	V77	0.062504	V235
3	0.016019	V68	0.031241	V458
4	0.015659	V53	0.029214	V203
5	0.014481	V44	0.025526	V249
6	0.013948	V70	0.020607	V354
7	0.013459	V46	0.019154	V43
8	0.012141	V203	0.016458	V454
9	0.011764	V76	0.014755	V76

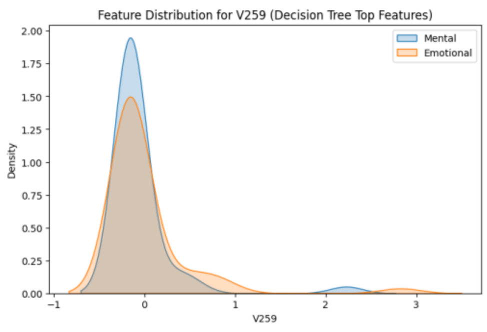
- Density Plots:

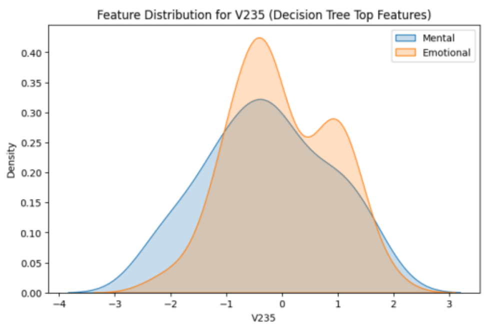










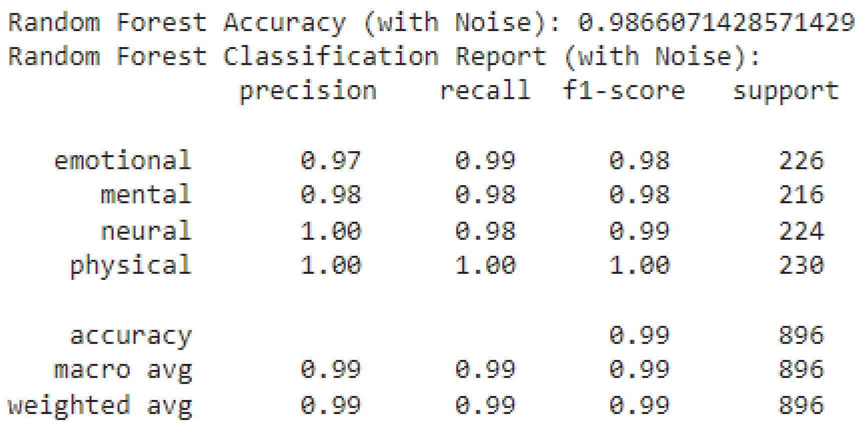


How Robust Are the Models to Noisy Data?

We added noise to 10% of the features and analysed the model's performance. Random Forest handled noise more effectively, with its accuracy only slightly reduced from 98.7% to 98.6%. Decision Tree, however, showed a more significant drop in performance, with its accuracy decreasing from 92.9% to 89.1%.

Plots:

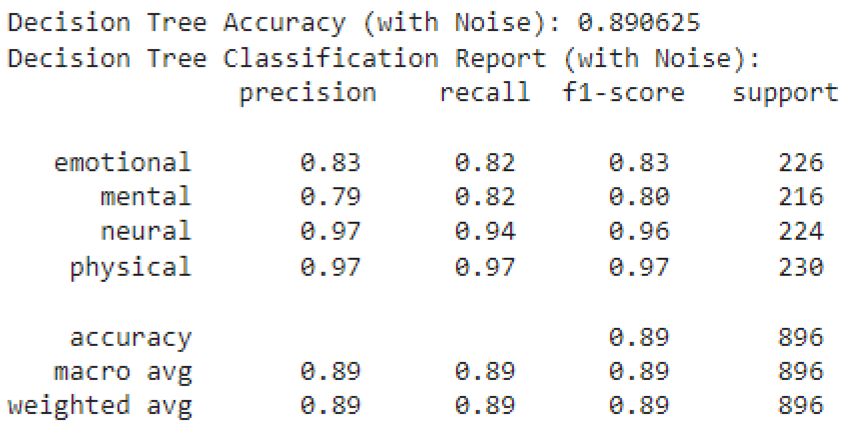
- Accuracy and Classification Reports with Noise:



```
Random Forest Accuracy (with Noise): 0.9866071428571429
Random Forest Classification Report (with Noise):
              precision    recall  f1-score   support

   emotional       0.97      0.99      0.98       226
      mental       0.98      0.98      0.98       216
      neural       1.00      0.98      0.99       224
    physical       1.00      1.00      1.00       230

    accuracy                           0.99       896
   macro avg       0.99      0.99      0.99       896
weighted avg       0.99      0.99      0.99       896
```

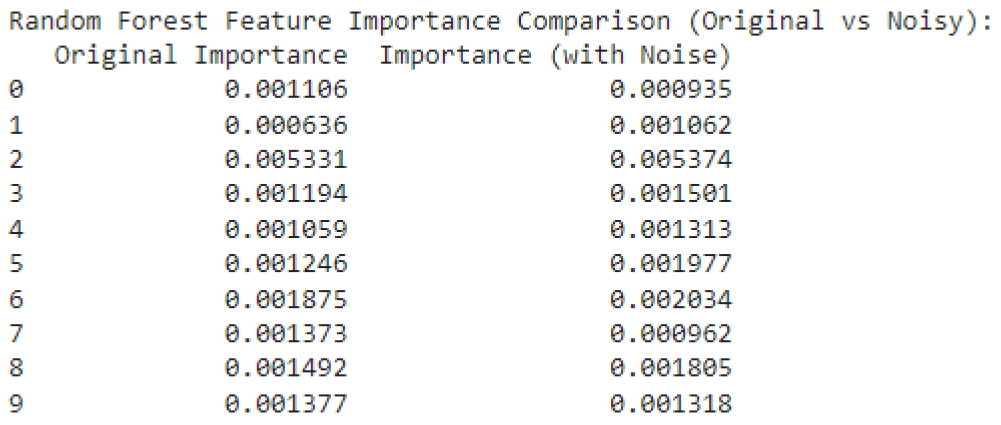


```
Decision Tree Accuracy (with Noise): 0.890625
Decision Tree Classification Report (with Noise):
              precision    recall  f1-score   support

   emotional       0.83      0.82      0.83       226
      mental       0.79      0.82      0.80       216
      neural       0.97      0.94      0.96       224
    physical       0.97      0.97      0.97       230

    accuracy                           0.89       896
   macro avg       0.89      0.89      0.89       896
weighted avg       0.89      0.89      0.89       896
```

- Feature Importance Comparison (With vs. Without Noise):



```
Random Forest Feature Importance Comparison (Original vs Noisy):
   Original Importance  Importance (with Noise)
0             0.001106                 0.000935
1             0.000636                 0.001062
2             0.005331                 0.005374
3             0.001194                 0.001501
4             0.001059                 0.001313
5             0.001246                 0.001977
6             0.001875                 0.002034
7             0.001373                 0.000962
8             0.001492                 0.001805
9             0.001377                 0.001318
```

```
Decision Tree Feature Importance Comparison (Original vs Noisy):
   Original Importance  Importance (with Noise)
0             0.000000                 0.000000
1             0.000000                 0.000000
2             0.002096                 0.002096
3             0.000000                 0.000000
4             0.000714                 0.001116
5             0.000000                 0.000000
6             0.004165                 0.004165
7             0.004960                 0.000000
8             0.006086                 0.002558
9             0.000000                 0.000000
```

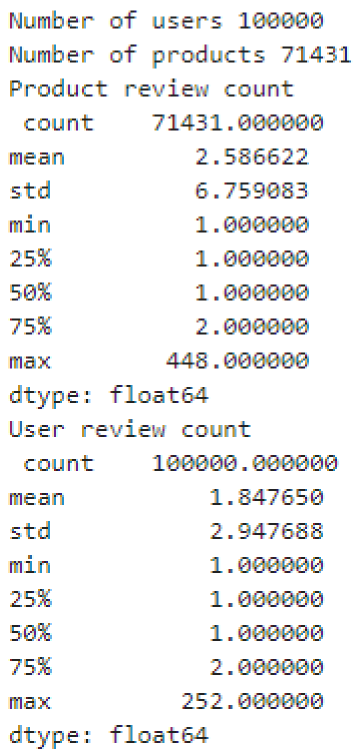
2. Lab 2 – Recommender systems

Introduction

In this section of the report, we will analyze and compare multiple recommendation models KNNBasic, BaselineOnly, and SVD for predicting user ratings on Amazon Electronics products. The dataset consists of user-product interactions where users rate various electronics, and we aim to build a collaborative filtering-based recommender system to predict ratings for unseen products.

What is the structure of the ratings matrix? Distribution of ratings, and is it long-tailed?

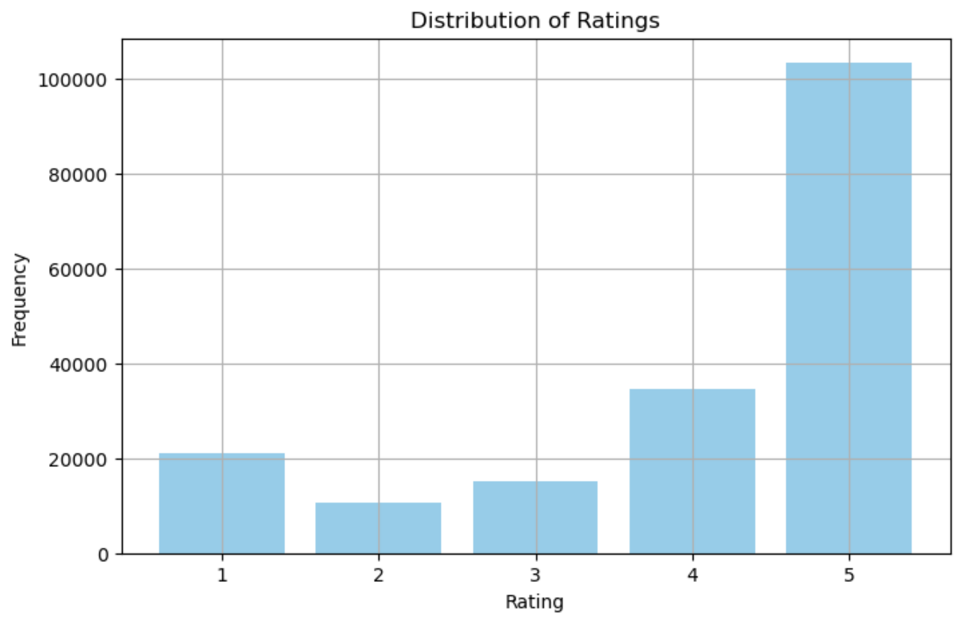
For this question, we explore the structure of the ratings matrix and examine the distribution of ratings in the dataset. We will aim to understand how sparse the matrix is and whether the distribution of ratings follows a long-tailed pattern, which is typical in user-product interaction datasets.



```
Number of users 100000
Number of products 71431
Product review count
 count    71431.000000
mean          2.586622
std           6.759083
min           1.000000
25%           1.000000
50%           1.000000
75%           2.000000
max         448.000000
dtype: float64
User review count
 count    100000.000000
mean           1.847650
std            2.947688
min            1.000000
25%            1.000000
50%            1.000000
75%            2.000000
max          252.000000
dtype: float64
```

Matrix Structure:

- Users: The dataset consists of 100,000 unique users.
- Products: There are 71,431 unique products.
- Ratings: The dataset contains 184,765 total ratings, indicating that the majority of users have rated only a small subset of products. This results in a sparse matrix, which is common in recommender systems where most users interact with only a limited number of products.



Ratings Distribution:

- The ratings range from 1 to 5, with 5 star ratings being the most frequent.
- The distribution is heavily skewed toward higher ratings, with the following breakdown:
  - 5 stars: The most common rating, with over 100,000 ratings.
  - 4 stars: The next most common rating.
  - Lower ratings (1, 2, and 3 stars) are much less frequent.

Long-Tailed Distribution:

- This dataset shows a long-tail pattern: A small number of products receive a large number of ratings, while most products receive very few ratings. The average number of ratings per product is only 2.59, indicating that the majority of products have limited interactions.

How many reviews do the most popular electronics products have? What about the customers and the number of reviews each gives?

Now we investigate the number of reviews for the most popular products and explore the distribution of user activity and how many reviews each user provides. Understanding these patterns will help us in identifying common challenges in recommender systems, such as the long-tail effect and user activity imbalances.

Product Reviews:

- Most Popular Product: The most reviewed product in the dataset has 448 reviews.

- Average Reviews per Product: On average, each product has been reviewed 2.59 times. This indicates that while a few products receive many reviews, most products receive only a small number of reviews.
- Long-Tail Distribution: Most products in the dataset have 1 or 2 reviews, with only a small subset of products receiving hundreds of reviews.

Customer Review Patterns:

- Most Active User: The most active user has reviewed 252 products.
- Average Reviews per User: The average user has given 1.85 reviews, and the median number of reviews per user is 1. This might mean users have only interacted with a few products, creating a sparsity problem where most users only rate a small number of items.
- User Activity Imbalance: Like products, user activity follows a long-tail distribution. A small number of users are highly active, while most users contribute only 1 or 2 reviews.

What is the best accuracy that you can get (RMSE)? How do models compare in terms of errors?

In this question, we compare the performance of the models: KNNBasic, BaselineOnly, and SDV. The models were evaluated based on their prediction accuracy, measured using RMSE.

Model parameters:

KNNBasic: 'name': 'cosine', 'user_based': False.

BaselineOnly: 'method': 'als', 'n_epochs': 5, 'reg_u': 12, 'reg_i': 5.

SVD: n_factors=2, n_epochs=100, biased=True, lr_all=0.001.

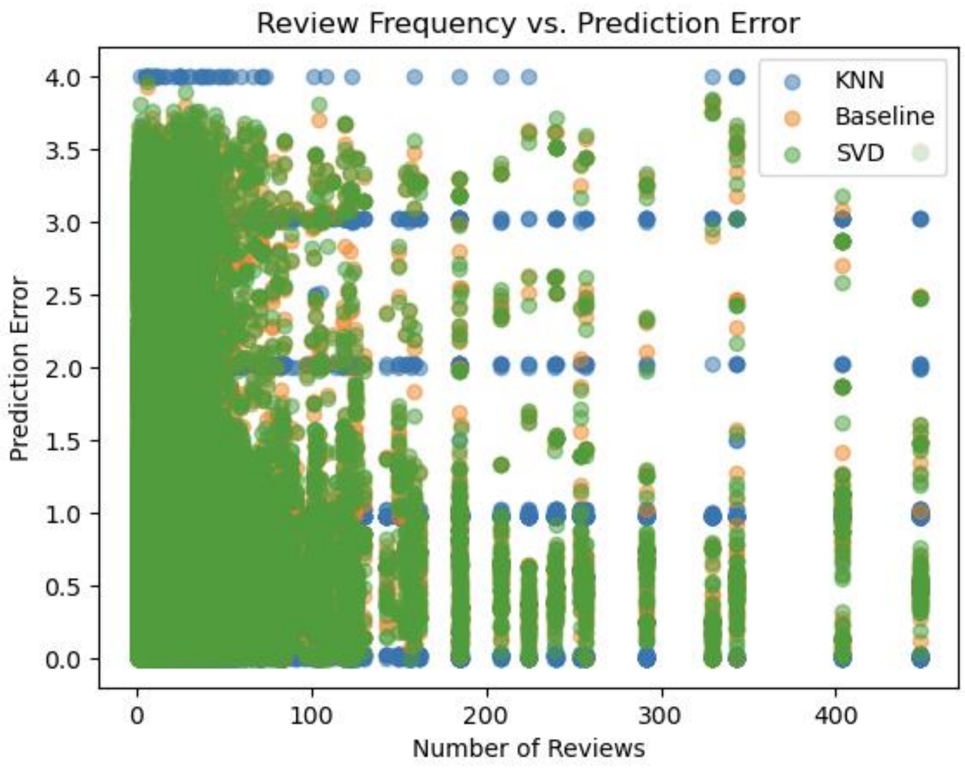
Model Performance (RMSE):

Model	RMSE
KNNBasic	1.3784
BaselineOnly	1.3309
SVD	1.3281

- SVD (Singular Value Decomposition) achieved the lowest RMSE of 1.3281, making it the most accurate model in this study. SVD captures the hidden relationships between users and items by decomposing the user-item interaction matrix into latent factors, leading to better predictions, especially for sparse data.
- BaselineOnly had a slightly higher RMSE of 1.3309, which incorporates bias terms for both users and items but does not capture the complex latent relationships between them as effectively as SVD does.
- KNNBasic (Cosine Similarity) performed the worst with an RMSE of 1.3784, primarily because it relies on item-item similarities.

Cold Start Problem:

- Cold Start Issue: KNNBasic is more prone to errors for products with few reviews, as it relies heavily on similarities between items. On the other hand, SVD manages the cold start problem better because it models the latent features of users and products, allowing it to make predictions even with limited interactions.



Model Comparisons:

- KNNBasic: Struggles with sparse data, relies on item-item similarities, and has higher prediction errors for products with fewer reviews.
- BaselineOnly: Adds bias terms for users and products, offering better accuracy than KNNBasic but still falls short of SVD.
- SVD: Best model, capturing latent factors that explain user preferences and product characteristics, resulting in superior predictions.

Are there items/users that are consistently difficult or easy to predict for all algorithms?

For this question, we explore whether there are specific items or users that are consistently difficult or easy to predict across all models (KNNBasic, BaselineOnly, and SVD). Understanding these patterns can help in identifying outliers or areas where models may struggle to generalize.

Easy-to-Predict Items:

- Consistency Across Models: There are several products for which all models make perfect predictions (with 0 error). These products had consistent and predictable ratings across different users.
- Characteristics of Easy Items: These easy-to-predict products are popular electronics with broad appeal, leading to consistent high ratings across the user base. Their predictability indicates that users tend to rate them similarly, regardless of the model used.

Difficult-to-Predict Items:

- High Prediction Errors: Some products had high prediction errors across all models (errors close to 4). These are products that were consistently hard to predict, due to inconsistent or polarizing ratings from users.
    - Examples include products like B00TYL27M and B00D029NNA, which showed high errors across all models (plot is below for top 10 difficulty to predict).
- Variable or Polarizing Ratings: The difficult-to-predict items have mixed reviews, with some users rating them very highly while others rate them poorly. This variability makes it difficult for all models to accurately predict the ratings.

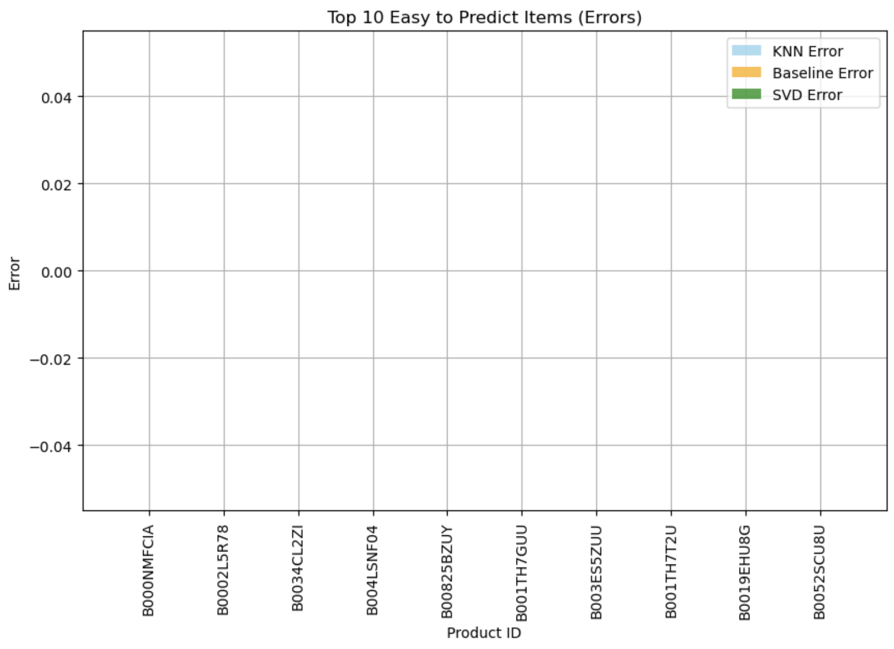
Consistent Errors for Users:

- Like items, certain users have ratings that are consistently harder to predict across all models. These users may have unique preferences or rating behavior that does not align well with other users, making it difficult for collaborative filtering techniques to accurately predict their preferences.

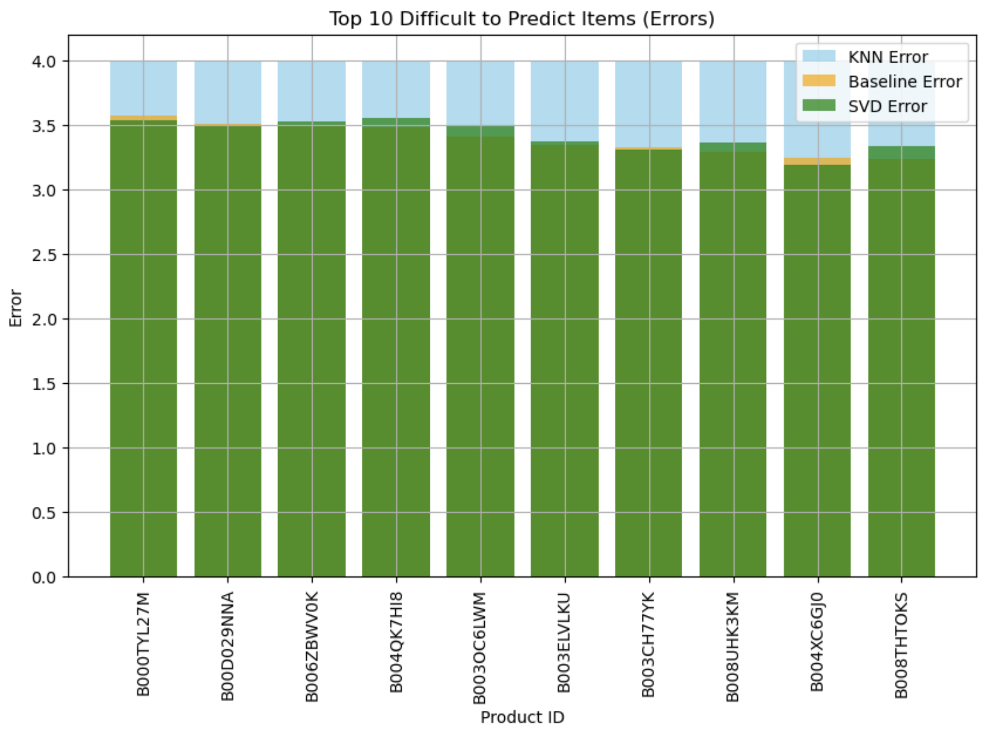
Visualization of Easy and Difficult Predictions:

We created two bar charts to illustrate these findings:

1. Top 10 Easy-to-Predict Items: This plot shows items with zero prediction errors across all models.



2. Top 10 Difficult-to-Predict Items: This plot shows items with the highest prediction errors across all models, indicating products that are challenging to predict due to mixed or inconsistent ratings.



Additional Explorations

In this section, we delve into additional analyses that provide further insights into model performance, particularly around the cold start problem and the latent factors from the SVD model.

Cold Start Problem

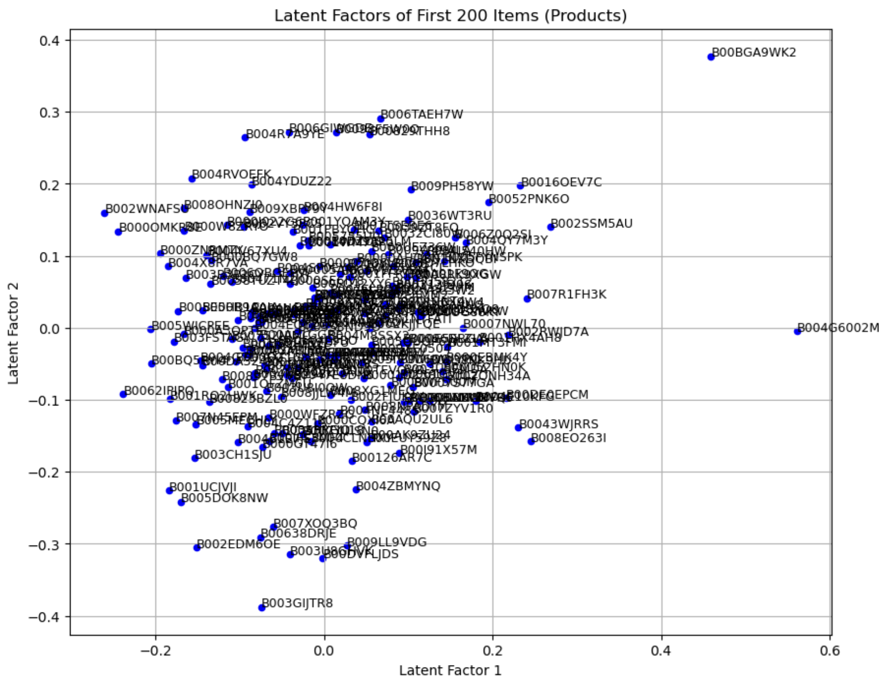
The cold start problem refers to the challenge of making accurate recommendations for products or users that have very few ratings. This problem is especially common in collaborative filtering systems where predictions rely on historical user-product interactions.

- Impact on Model Performance:
    - KNNBasic struggles significantly with cold start scenarios because it depends on the availability of similar items for comparison. When a product has only a few ratings, it becomes difficult for KNNBasic to find dependable neighbors, leading to high prediction errors.
    - SVD, on the other hand, manages cold start situations better due to its latent factor approach. Even with limited data, SVD can make reasonable predictions by leveraging the underlying patterns in the user-item interaction matrix.
- Visualization: We created a scatter plot of Review Frequency vs. Prediction Error, which demonstrates that products with fewer reviews tend to have higher errors. KNNBasic shows the most significant struggle with these products, while SVD performs better in low-data scenarios.

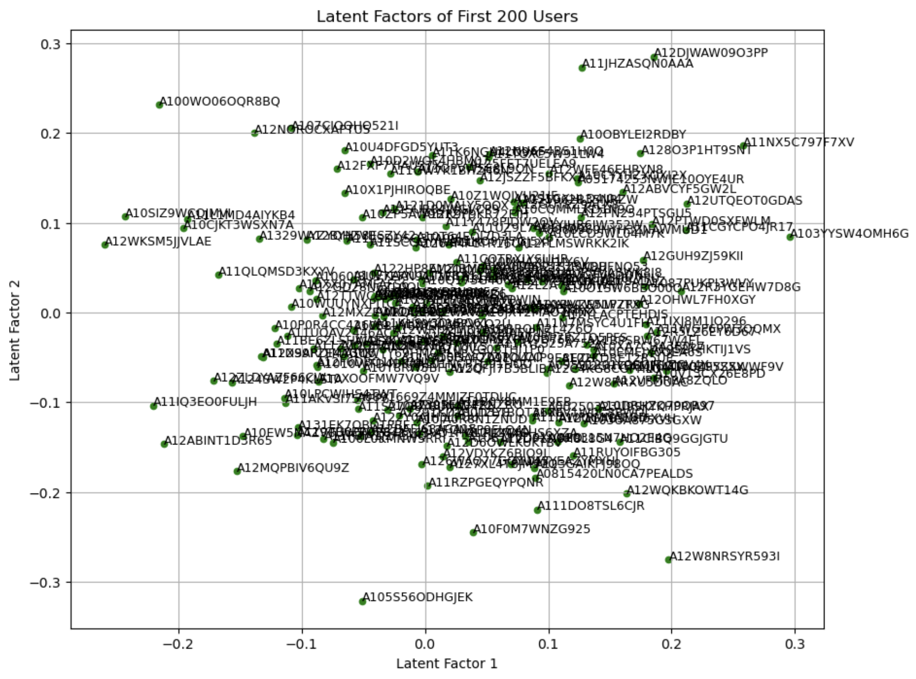
Latent Factor Analysis (SVD)

The SVD model is a matrix factorization technique that decomposes the user-item interaction matrix into latent factors. These latent factors capture hidden relationships between users and products, enabling the model to make more accurate predictions.

- Latent Factors for Products:
    - Products that are closer to each other in the latent space share similar characteristics or receive similar ratings from users. For example, products that are widely popular and well-rated will cluster together.
    - Outliers: Some products, such as B00GA9WK2 and B00456002M, are positioned far from the center, indicating that they may be niche products or have unusual rating patterns that set them apart from others.



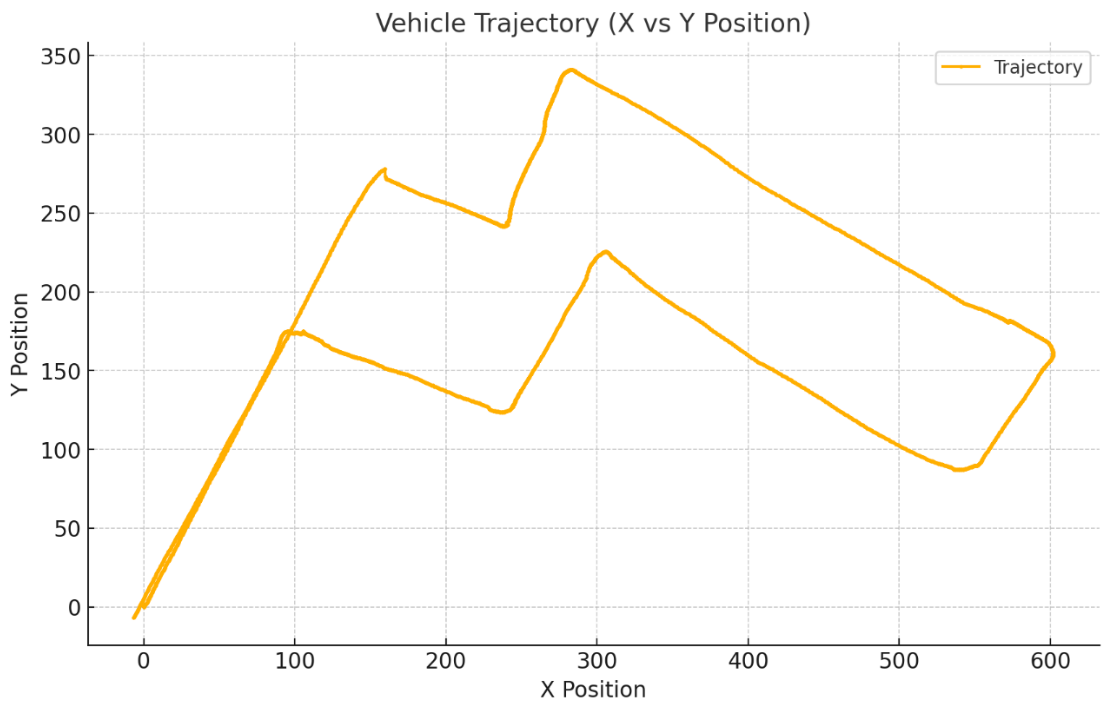
- Latent Factors for Users:
    - Users with similar rating behaviors are positioned closer to each other in the latent space. These users tend to rate similar products, similarly, making their preferences easier to predict.
    - Outliers: Users such as A12DWVAW9093PP and A12W8NRSYR593I are positioned farther away from the main cluster, indicating unique or extreme preferences, potentially making their ratings harder to predict.

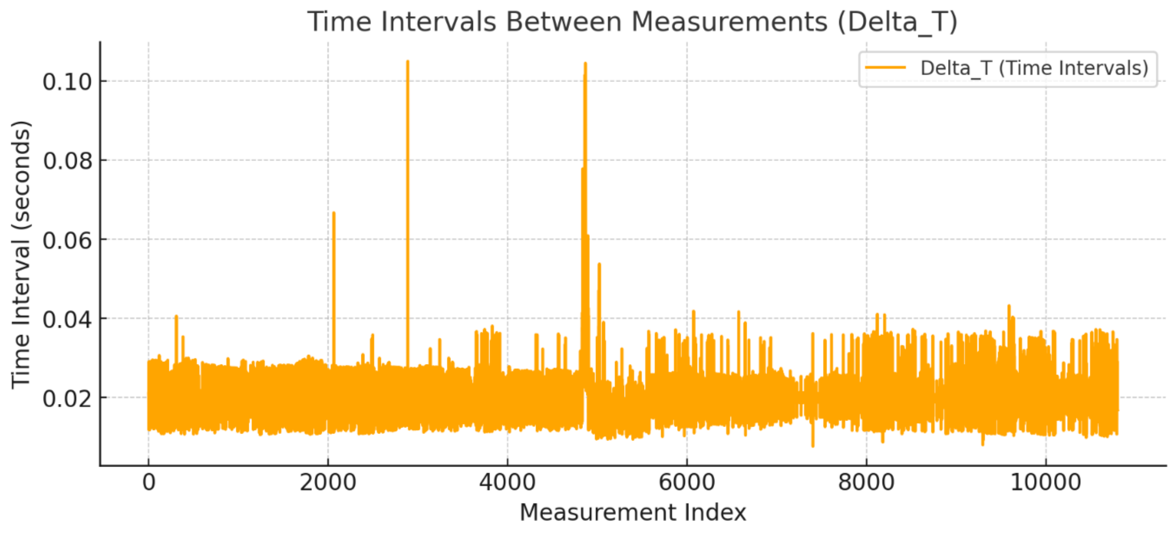


3. Lab 3

Introduction

In this section of the report, we wil implement and analyze the performance of an EKF to estimate the state of a vehicle using the noisy GPS position measurements. Our analysis will include sensitivity to initializations, impact of process noise Q and measurement noise R. Our dataset consisted of GPS measurements of a vehicle's x and y positions and its time intervals delta T.





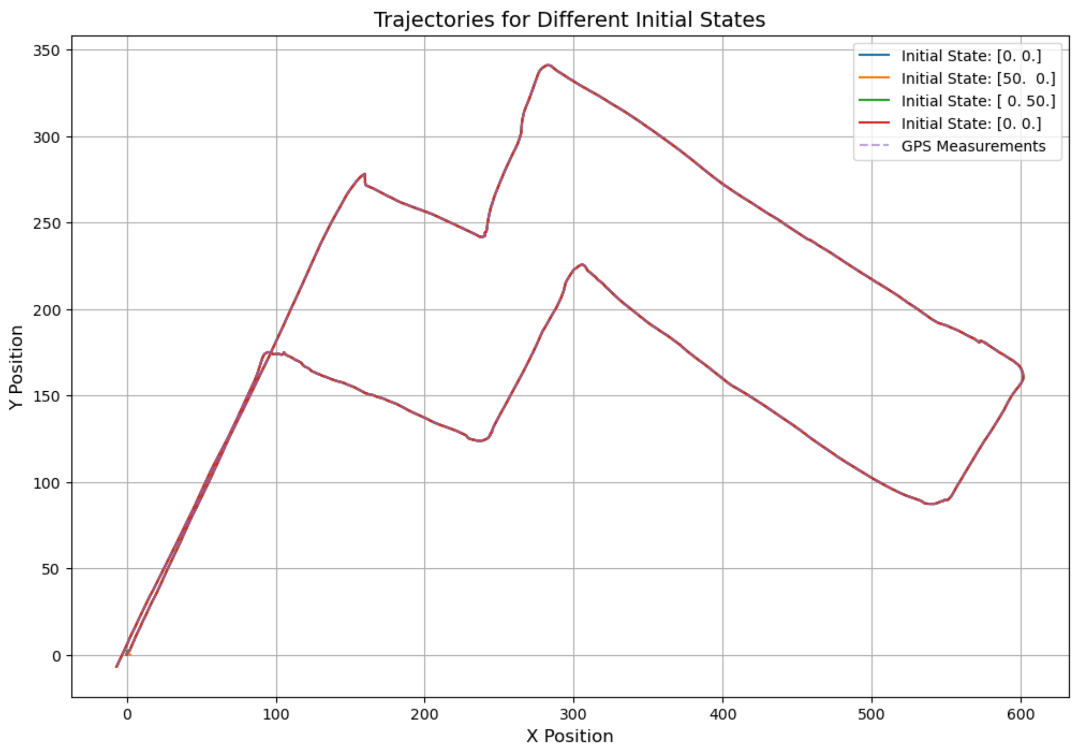
Sensitivity to initialization

Now we will evaluate how the EKF performs under different initial conditions focusing on variations in the initial state vector and changes in the initial uncertainty.

This will helps us understand how the EKF handles inaccuracies in the initial assumptions and whether it converges effectively despite these variations.

We will test the EKF's sensitivity to variations in the initial state vector x_0, which includes x,y positions, heading ψ, and velocity v.
The initiall state variations are

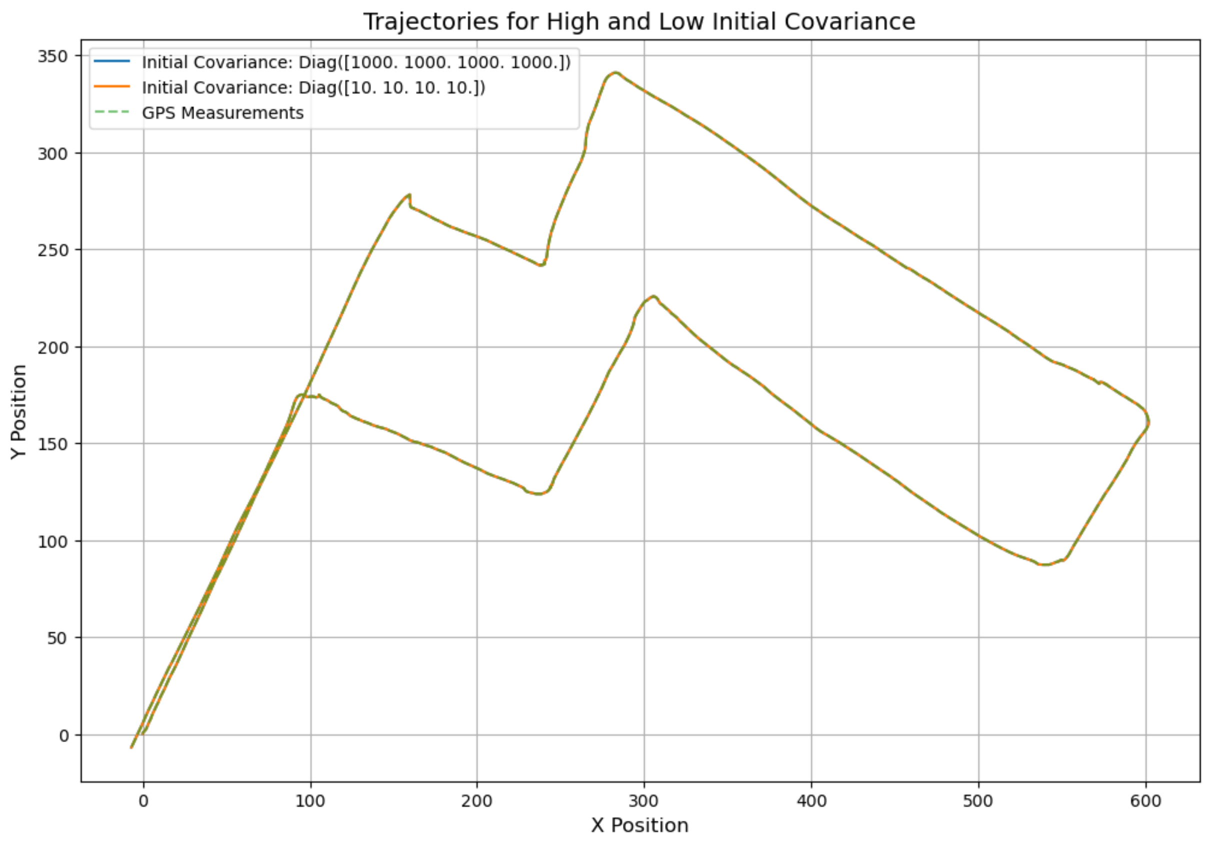
- Default initialization: $[x_0,y_0,\psi_0,v_0]=[0,0,0,0]$.
- Offset Positions: $[x_0,y_0]+[50,50]$.



Results:

- The EKF showed robust performance across all initial state variations.
- Convergence was achieved within a few steps, even with large offsets in the initial state or non-zero heading values.
- Deviations were higher initially but diminished quickly as the filter incorporated measurements.

What happens when the initial covariance changes?
The initial uncertainty (P_0) affects how much the EKF trusts its initial state. We evaluated:
High Uncertainty: $P_0 = \text{diag}([1000,1000,1000,1000])$.
Low Uncertainty: $P_0 = \text{diag}([10,10,10,10])$.



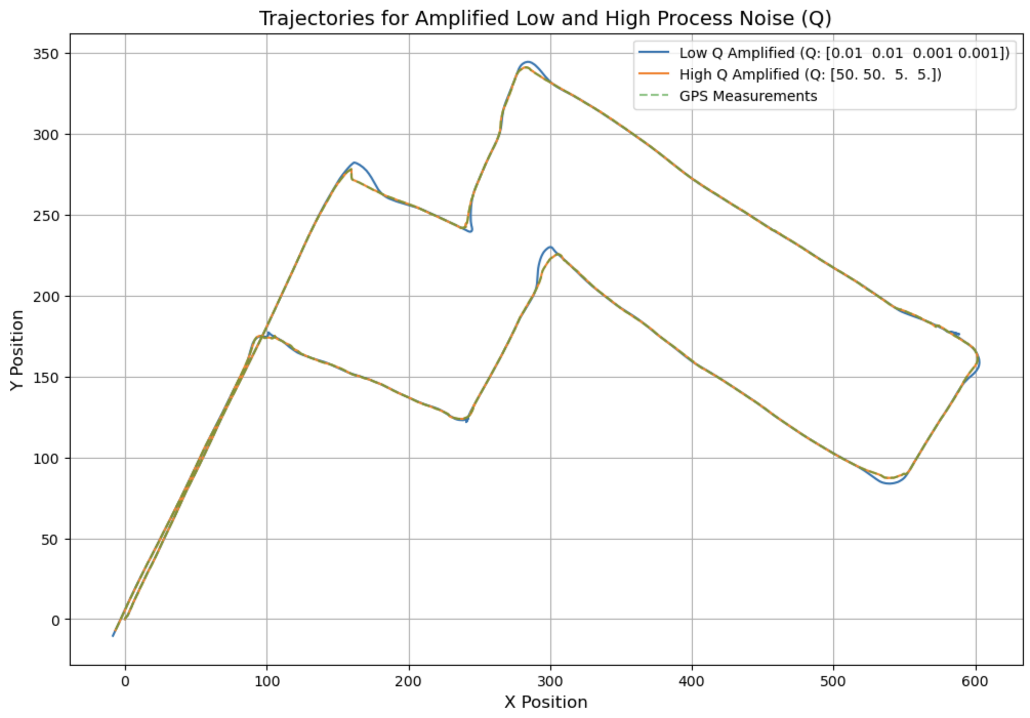
Sensitivity to Q and R covariance matrices

For this analysis, we investigate how changes in the process noise (Q) and measurement noise (R) covariance matrices affect the performance of EKF. We also analyze residuals and Kalman gains to better understand the balance between trusting the motion model versus relying on sensor measurements.

Q changes:
We analyzed the EKF performance with amplified values to better observe its effect:

- Low Q: diag([0.01,0.01,0.001,0.001])
- High Q: diag([50,50,5,5])

Results:

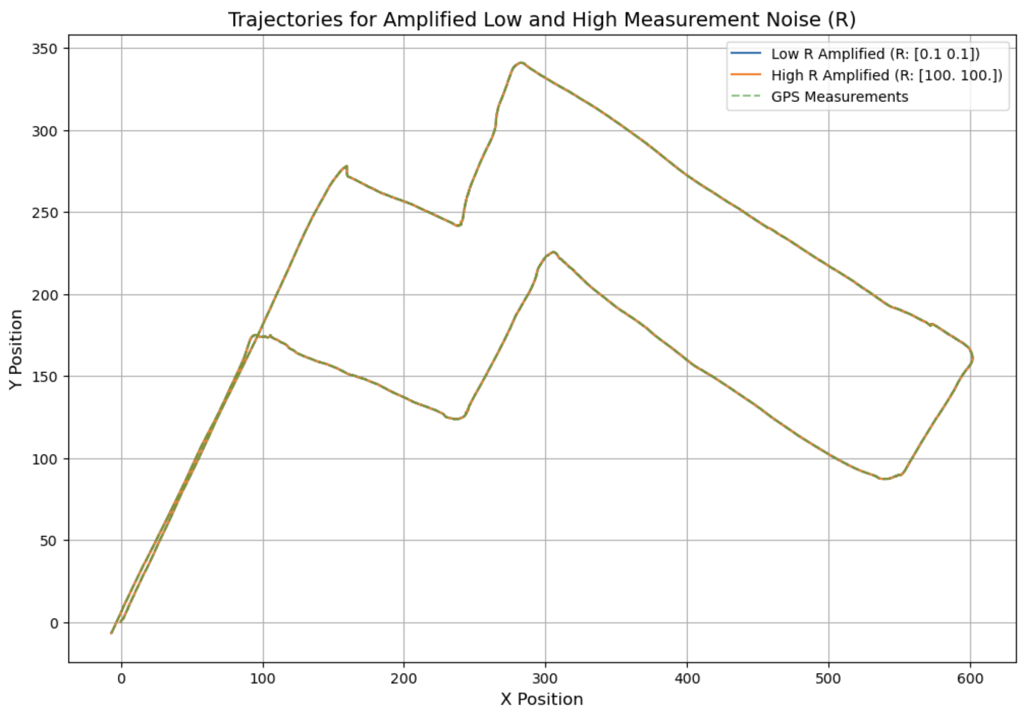


- Low Q:
  - Trajectories for low Q shows slight divergences when the measurements were noisy.
- High Q:
  - The filter relies more on the motion model, leading to a smoother trajectory with no visible divergences.

R changes:
We evaluated:

- Low R: diag([0.1,0.1])
- High R: diag([100,100])

Results:



- Low R:
  The filter discounts noisy measurements, resulting in a smoother trajectory.
- High R:
  The filter discounts noisy measurements, resulting in a smoother trajectory.

Robustness against "Kidnapping" (Missing data)

In this section, we evaluate the robustness of the EKFwhen some GPS measurements are removed. This simulates real-world scenarios where sensor data may at time be unavailable or corrupted.

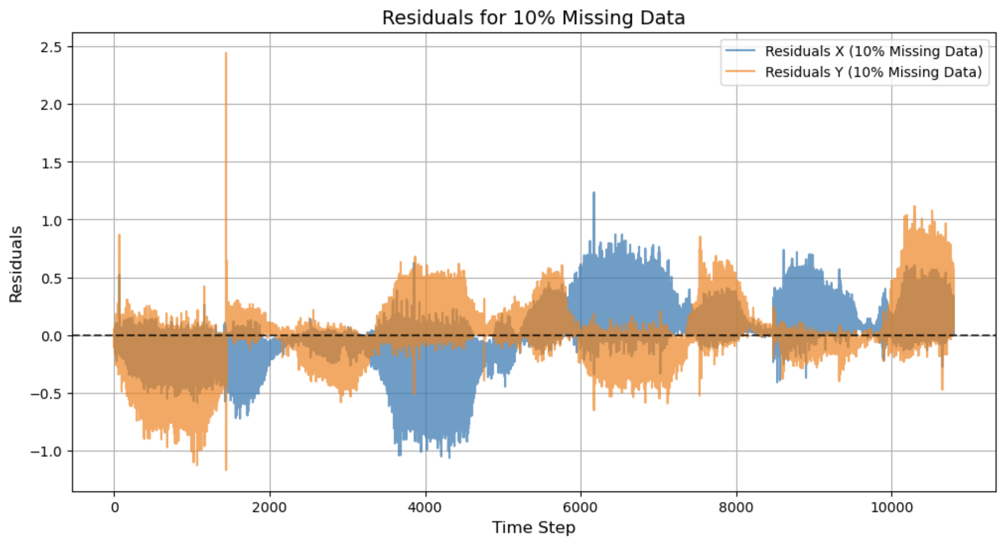
To evaluate the EKF's ability to recover from missing data, we randomly removed different percentages of GPS measurements from the dataset:

1. Randomly removed 10% of GPS measurements.
2. Randomly removed 30% of GPS measurements.
3. Randomly removed 50% of GPS measurements.

The EKF relies on the motion model to interpolate through gaps, and we analyze its ability to estimate the trajectory and recover states.

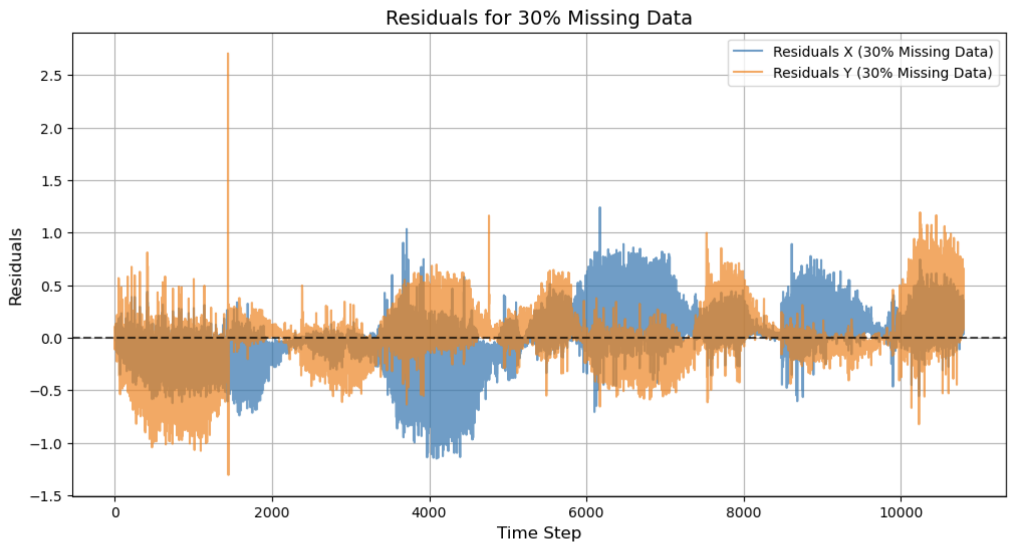
Results:
Residuals (x, y) measure the difference between the EKF's predictions and the actual GPS measurements. Residual analysis highlights the EKF's accuracy under different levels of missing data.

- 10% Missing Data:



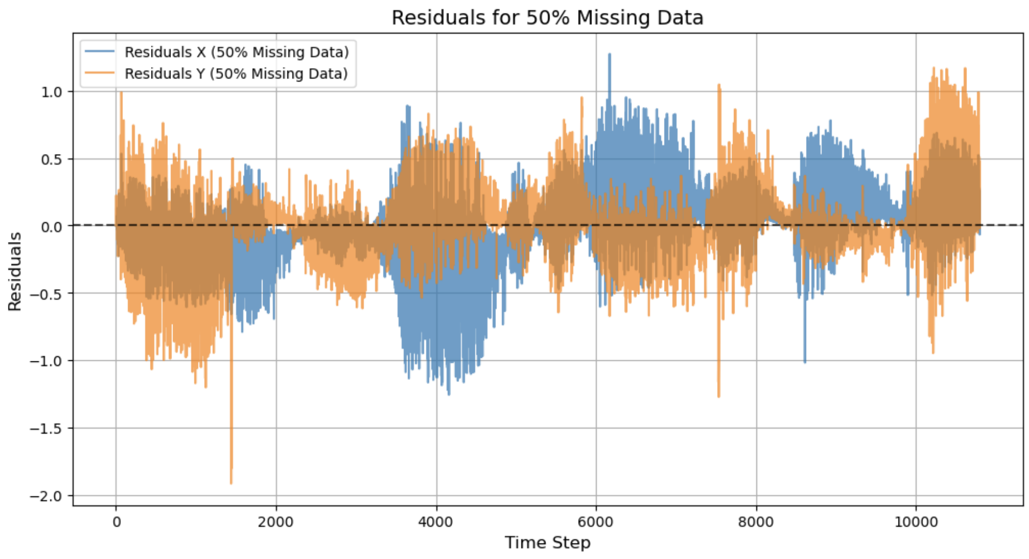
- Residuals remain small and stable, indicating accurate interpolation over small gaps.
- Occasional spikes reflect moments when the EKF struggles to settle gaps with measurements.

- 30% Missing Data:



- Residuals increase slightly, particularly in regions where the EKF interpolates over larger gaps.
- Despite more missing data, residuals remain controlled, showing the EKF's robustness.

- 50% Missing Data:



- Residuals are larger and more variable, particularly during extended gaps or rapid trajectory changes.
- The EKF's reliance on the motion model leads to smoother predictions but less accurate alignment with the ground truth.

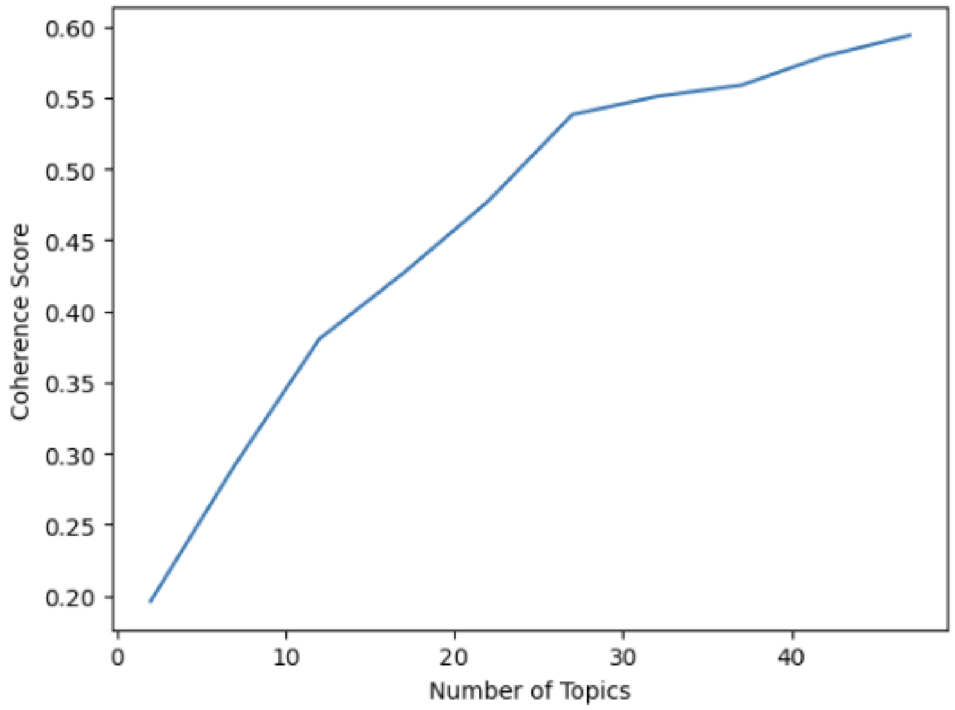
4. Lab 4

Introduction

In this section of the report, we use LDA to analyze topics within a dataset of fake and real news. We will try to address the eight key questions to understand the topic coherence, interpretability, and differences in topic distributions between fake and real news. The model's focus will help us find insights on major topics and compare the insights found from titles and full texts. We will use tools such as gensim and coherence scores to help evaluate the quality and stability of the topics.

What is a reasonable/optimal number of topics for the entire dataset?

We evaluated coherence scores across a range of topics (from 2 to 40) to identify the optimal number of topics. The coherence score measures how interpretable the topics are based on their word combinations.



- The plot shows a steady improvement in coherence scores until around 35 topics, where the scores stabilize. This elbow point suggests that 35 topics provide the best balance between prec and coherence.

Consider the case of a very small number of topics (e.g., 3). Are the inferred topics reasonable/meaningful?

We trained an LDA model with only 3 topics to evaluate whether the results are meaningful. The top keywords for each topic are as follows:

Topic	*Keyword*
1	comment, email, police, get, kill, show, say, year, go, child
2	trump, new, say, campaign, time, first, woman, man, video, watch
3	trump, breitbait, election, call, trump, report, vote, say, day

What We Found:

- Topic 0 relates to crime and societal issues.
- Topic 1 focuses on Trump and political events.
- Topic 2 combines election-related topics.

Although meaningful at a high level, these topics are too broad to provide detailed insights.

Consider the case of a very large number of topics (e.g., 1000). Are the inferred topics reasonable/meaningful?

We trained an LDA model with 1000 topics to evaluate whether the results are meaningful. The top keywords for each topic are as follows:

Topic	*Keyword*
504	delusional, drench, salesforce, shareholder, nomi, prin, memory, ww2, verify, amnesty
535	delusional, drench, salesforce, shareholder, nomi, prin, memory, ww2, verify, amnesty
813	delusional, drench, salesforce, shareholder, nomi, prin, memory, ww2, verify, amnesty
622	delusional, drench, salesforce, shareholder, nomi, prin, memory, ww2, verify, amnesty
12	delusional, drench, salesforce, shareholder, nomi, prin, memory, ww2, verify, amnesty
155	delusional, drench, salesforce, shareholder, nomi, prin, memory, ww2, verify, amnesty
348	delusional, drench, salesforce, shareholder, nomi, prin, memory, ww2, verify, amnesty
716	delusional, drench, salesforce, shareholder, nomi, prin, memory, ww2, verify, amnesty
148	delusional, drench, salesforce, shareholder, nomi, prin, memory, ww2, verify, amnesty
922	delusional, drench, salesforce, shareholder, nomi, prin, memory, ww2, verify, amnesty

What We Found:
The topics are fragmented and repetitive, with similar keywords appearing in multiple topics. This level of precision does not add meaningful insights and makes the topics less interpretable.

What are the topic distributions for fake and true news?

We trained separate LDA models for fake and real news (using 35 topics) and compared their topic distributions.

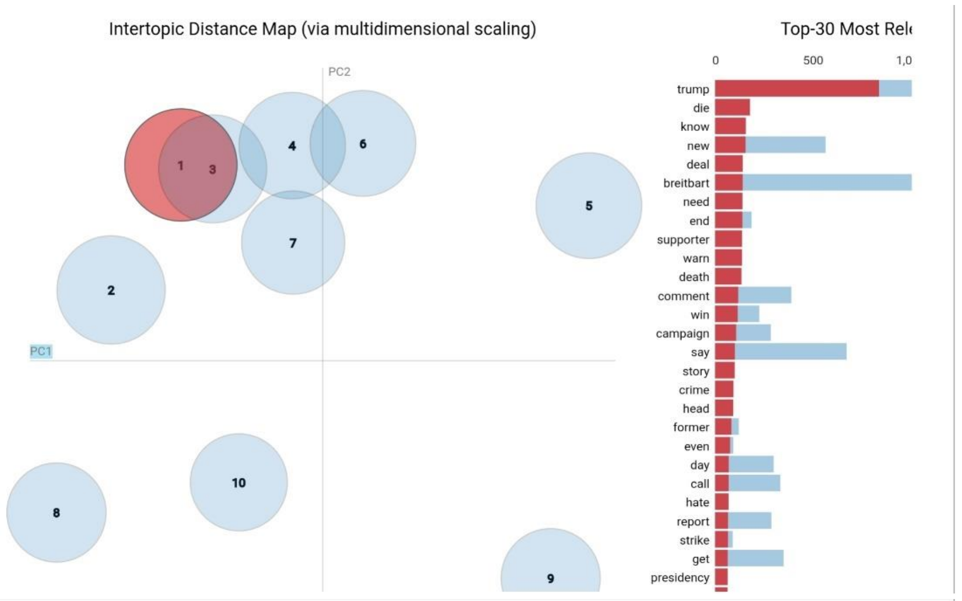
Category	*Dominant Topics*
Fake News	trump, election, scandal, email, fraud
Real News	government, policy, health, economy, law

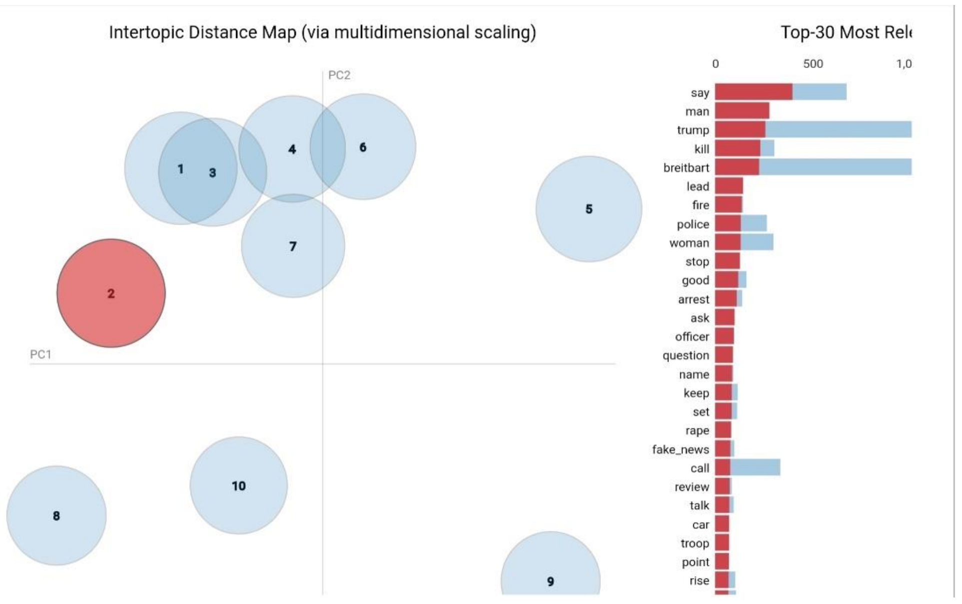
What We Found:
Fake news tends to focus on sensational topics like scandals, fraud, and election controversies, often featuring emotionally charged keywords. In contrast, real news covers a broader range of topics, including governance, policy, and social issues, indicating more structured reporting.

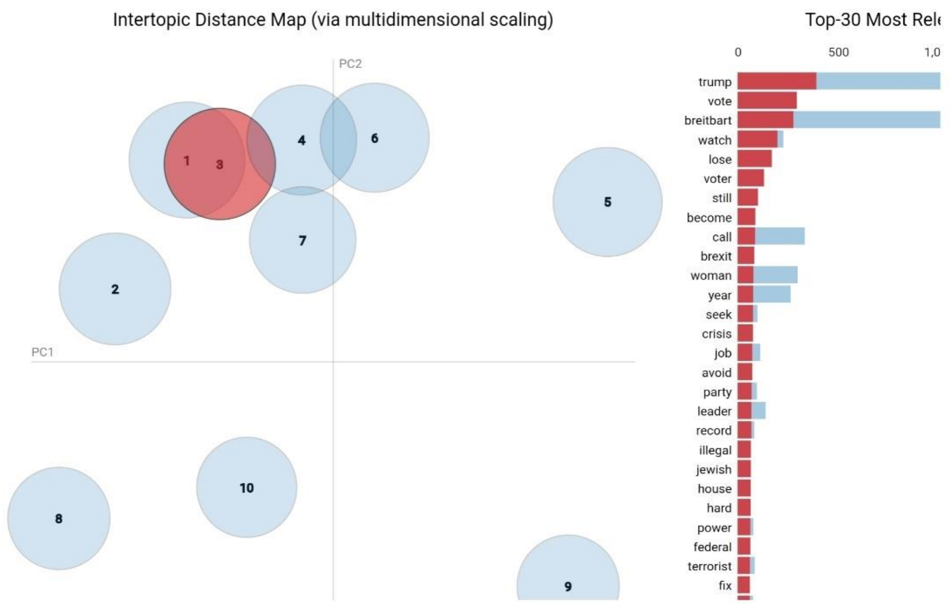
Can you guess what are the major topics that were reported? Where do the fake news stand in relation to all topics?

The major topics across the dataset were identified:

- Topic 1: trump, die, know, new, deal, breitbait.
- Topic 2: say, man, trump, kill, breitbait, lead, fire.
- Topic 3: trump, vote, breitbait, watch, lose, voter.


Intertopic Distance Map (via multidimensional scaling)
PC1
PC2
1
2
3
4
5
6
7
8
9
10
Top-30 Most Rel
0
500
1,0
trump
die
know
new
deal
breitbart
need
end
supporter
warn
death
comment
win
campaign
say
story
crime
head
former
even
day
call
hate
report
strike
get
presidency


Intertopic Distance Map (via multidimensional scaling)
PC1
PC2
1
2
3
4
5
6
7
8
9
10
Top-30 Most Rel
0
500
1,0
say
man
trump
kill
breitbart
lead
fire
police
woman
stop
good
arrest
ask
officer
question
name
keep
set
rape
fake_news
call
review
talk
car
troop
point
rise


Intertopic Distance Map (via multidimensional scaling)
PC1
PC2
1
2
3
4
5
6
7
8
9
10
Top-30 Most Rel
0
500
1,0
trump
vote
breitbart
watch
lose
voter
still
become
call
brexit
woman
year
seek
crisis
job
avoid
party
leader
record
illegal
jewish
house
hard
power
federal
terrorist
fix

Give examples of some topics with the most representative real and fake news texts.

We identified representative texts for each topic using their highest topic contribution scores.

Topic	*Representative Text*
Fake News	Email scandal reveals election fraud details.
Real News	Government announces new education policy.

What We Found:
The representative texts align well with their corresponding keywords. For example:

- Fake news often focuses on scandals or conspiracies, as seen in the election fraud example.
- Real news provides more neutral and policy-focused content, like the education policy example.

Split the dataset into two parts: fake and real news. Run LDA for each part. Did this make a considerable difference in the optimal number of topics?

How texts correlate with the topics uncovered by using only titles?

We compared topics derived from titles with those from the full text. Here's a summary of the overlap:

Topic	*Dominant Keywords*
Titles	trump, vote, campaign, fraud
Full Texts	government, policy, economy, law

What We Found:
Title-based topics often align with high-level themes, focusing on broad issues like elections and scandals. Full-text topics are more detailed, covering nuanced aspects like specific policy issues or local events. Titles can capture general trends but lack the depth of full-text analysis.
s