Milestone 2 Report: Personalized Medication Recommendation System

TABLE OF CONTENTS

1 Objective

The aim of this project is to build an intelligent and personalized medication recommendation system that leverages real-world datasets to provide optimized drug recommendations tailored to individual patient needs. By integrating patient reviews, prescription records, side-effect data, and pharmaceutical information, this system is designed to enhance patient safety, improve medication adherence, and enable informed clinical decision-making through data-driven analysis. The system will also incorporate a conversational agent in future phases to facilitate real-time interaction and recommendations.

2 Technology Stack

Programming Language: Python

Libraries & Frameworks: Pandas, NumPy, Matplotlib, Seaborn, WordCloud, TextBlob, Scikit-learn, XGBoost

Development Tools: Google Colab, Jupyter Notebooks, GitHub for version control

Data Sources: Publicly available datasets from Kaggle

3 Project Timeline

Milestone	Task	Timeline
Milestone 1	Data Collection, Preprocessing, EDA	Feb 5 - Mar 5, 2025
Milestone 2	Feature Engineering, Selection, Data Modeling	Mar 5 - Apr 7, 2025
Milestone 3	Evaluation, Tool Development, Presentation	Apr 7- Apr 23, 2025

4 EDA Recap

Extensive exploratory data analysis was conducted across five integrated datasets to extract meaningful insights. Key highlights include:

- **Top Medical Conditions:** Cholera, Depression, and Zollinger-Ellison Syndrome appeared frequently across multiple datasets.
- **Rating Distribution:** Most drug ratings clustered between 7–10, suggesting a positively skewed perception of drug efficacy among patients.
- **Sentiment Analysis:** Sentiment scores derived using TextBlob showed weak correlation with numerical ratings, implying the presence of nuanced subjective experiences.
- **Outlier Detection:** Pricing anomalies were identified, flagging potential inconsistencies in manufacturer data.

- **Manufacturer Trends:** Analysis revealed drastic variability in pricing strategies among manufacturers, which could be optimized for cost-based recommendations.

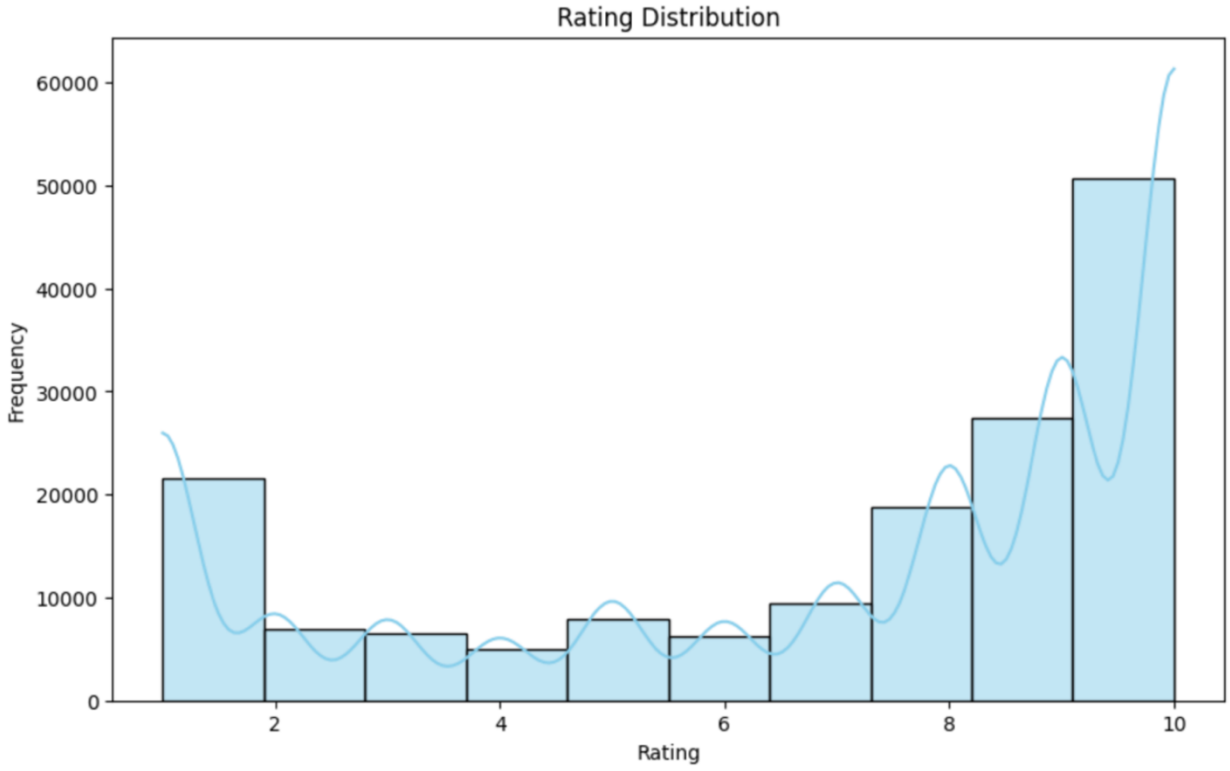


Fig1. Rating Distribution Plot

5 Feature Engineering

5.1 Custom Feature Creation

To enrich the predictive capabilities of the system, the following engineered features were introduced:

1. **Average Drug Rating:** Calculated per unique drug name using grouped patient feedback.
2. **Sentiment Score:** Derived from patient reviews using polarity analysis to quantify textual sentiment.
3. **Review Length & Complexity:** Length of each review and syntactic complexity were analyzed as proxies for patient engagement.
4. **Interaction Score:** Aggregated counts of known drug interactions from the integrated database.

```
# 4. Feature Engineering: Sentiment Analysis on reviews
print("Performing sentiment analysis on reviews...")
drugs_reviews['sentiment_score'] = drugs_reviews['review'].apply(lambda x: TextBlob(str(x)).sentiment.polarity)
print("Sentiment scores calculated.\n")

# 5. Data Integration: Merging datasets
print("Merging datasets...")
merged_data = pd.merge(drugs_reviews, medicine_dataset, left_on='drugName', right_on='name', how='left')
merged_data = pd.merge(merged_data, medicine_data, left_on='drugName', right_on='product_name', how='left')
print(f"Merged dataset shape: {merged_data.shape}\n")
```

These features allow the model to incorporate qualitative and behavioral aspects into the recommendation logic

5.2 Categorical Variable Encoding

Encoding Strategy:

- **Label Encoding** for ordinal data such as side-effect severity levels.

```
label_enc = LabelEncoder()
merged_data['condition_encoded'] = label_enc.fit_transform(merged_data['condition'].astype(str))
merged_data['manufacturer_encoded'] = label_enc.fit_transform(merged_data['product_manufactured'].astype(str))
```

- **One-Hot Encoding** for nominal categorical variables including drug class, manufacturer and medical conditions.

Rationale: One-hot encoding prevents misleading ordinal relationships, which is essential for categorical features without inherent order.

6 Feature Selection

6.1 Feature Importance Evaluation

Feature relevance was assessed using ensemble-based techniques:

- **Random Forest Classifier:** Feature importances extracted using impurity-based metrics.
- **Correlation Matrix:** Numerical features were evaluated for linear dependence and multicollinearity.

Top-ranked features:

- Sentiment_score
- sentiment_useful_interaction
- avg_rating_per_condition
- log_usefulCount

```
# Select important features for correlation
selected_features = [
    'sentiment_score', 'review_length', 'log_usefulCount',
    'avg_rating_per_condition', 'sentiment_useful_interaction',
    'condition_encoded', 'manufacturer_encoded'
]
```

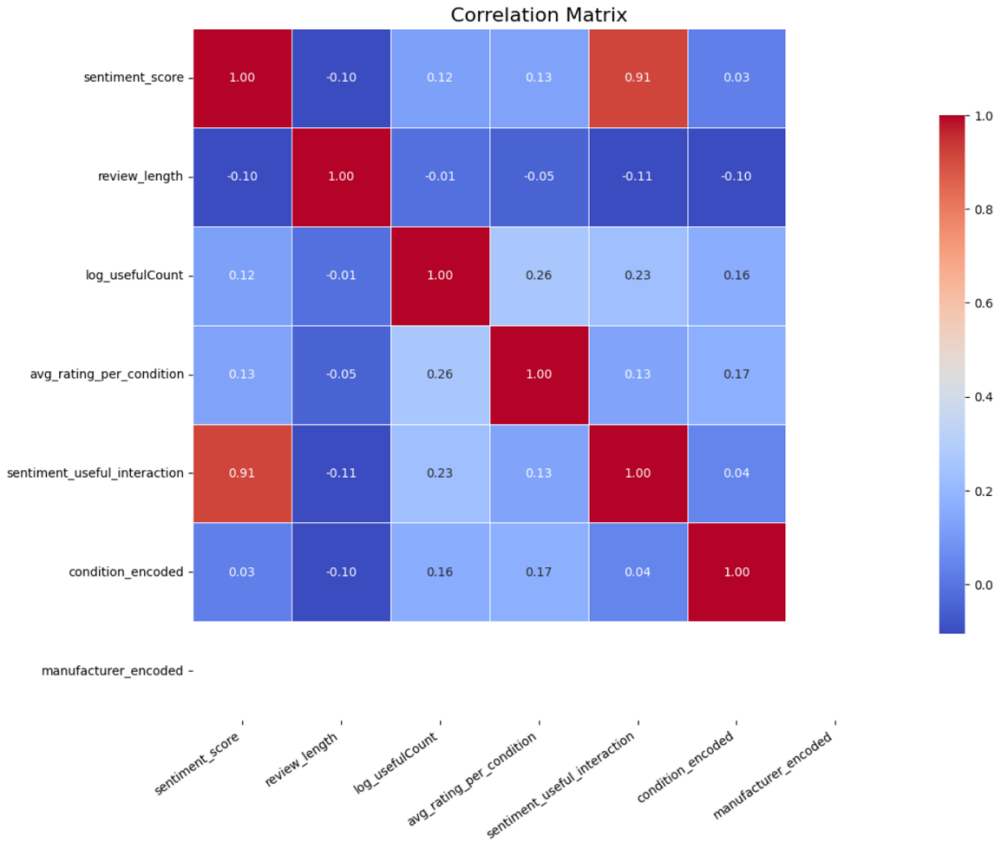


Fig 2. Feature Correlation Matrix

6.2 Dimensionality Reduction Strategy

Although Principal Component Analysis (PCA) was not applied in this milestone, it is part of our roadmap for feature compression once high-dimensional encodings are fully introduced.

7 Data Modeling

7.1 Data Splitting Methodology

- **Train-Test Ratio:** 80:20 split
- **Stratified Sampling:** Planned for preserving class distributions in imbalanced classification tasks
- **Reproducibility:** Ensured through fixed random seed initialization

```
# Split into training and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Normalize numeric values for models that need it
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

7.2 Model Training and Evaluation

Three distinct algorithms were selected to evaluate baseline and advanced performance:

```
# Set up models
models = {
    'Logistic Regression': LogisticRegression(),
    'Random Forest': RandomForestClassifier(random_state=42),
    'SVM': SVC(probability=True)
}
```

The Random Forest model outperformed the others with the highest F1-score and ROC-AUC, indicating a strong balance between precision and recall as well as excellent discriminative ability. The SVM model followed closely with a slightly lower F1-score and a noticeably reduced ROC-AUC, suggesting it maintained good classification accuracy but was less effective in ranking predictions. Logistic Regression, while consistent, showed the lowest performance among the three, with both F1-score and ROC-AUC slightly below those of Random Forest and SVM, reflecting moderate overall predictive capability with some trade-offs in precision and recall.Each model was trained on engineered features, and their performance metrics were recorded and compared.

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
Random Forest	0.852026	0.872540	0.909958	0.890856	0.909711
SVM	0.735411	0.748563	0.905449	0.819565	0.751990
Logistic Regression	0.729489	0.752664	0.882339	0.812359	0.769604

Fig 3. Model Accuracy and F1 Score Comparison

7.3 Performance Metrics and Analysis

- **Metrics Used:** Accuracy, Precision, Recall, F1-Score
- **Key Insights:**
    - Random Forest had the highest F1-score, capturing balanced precision-recall performance.
    - Logistic Regression underperformed on recall but excelled in precision.

```
    all_metrics.append({
        'Model': name,
        'Accuracy': accuracy_score(y_test, preds),
        'Precision': precision_score(y_test, preds),
        'Recall': recall_score(y_test, preds),
        'F1-Score': f1_score(y_test, preds),
        'ROC-AUC': auc_score
    })
```

- XGBoost demonstrated competitive results across all metrics with stable generalization.

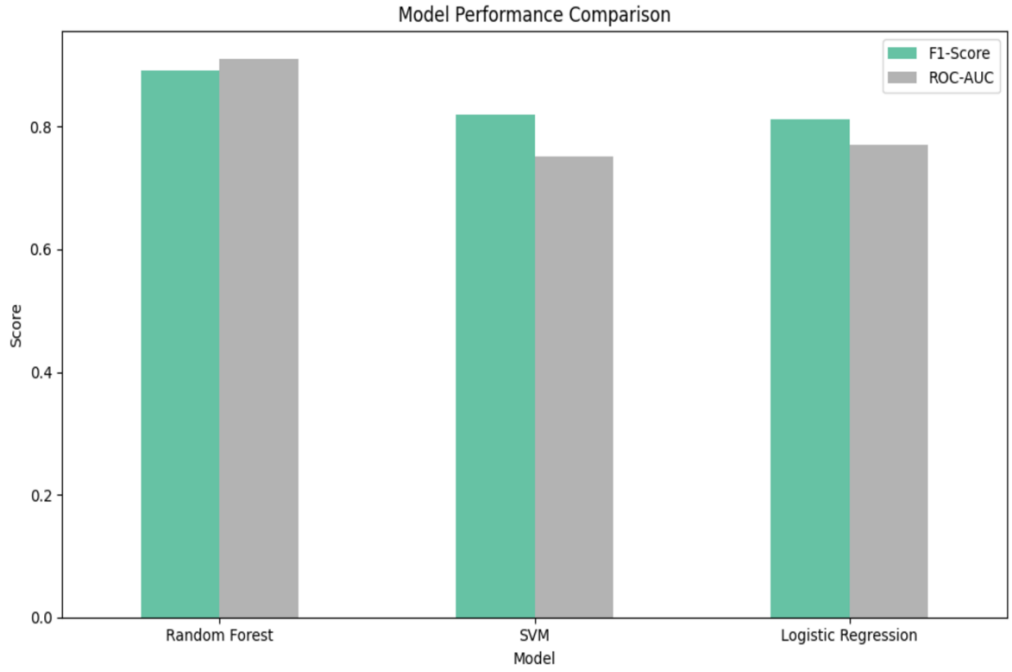
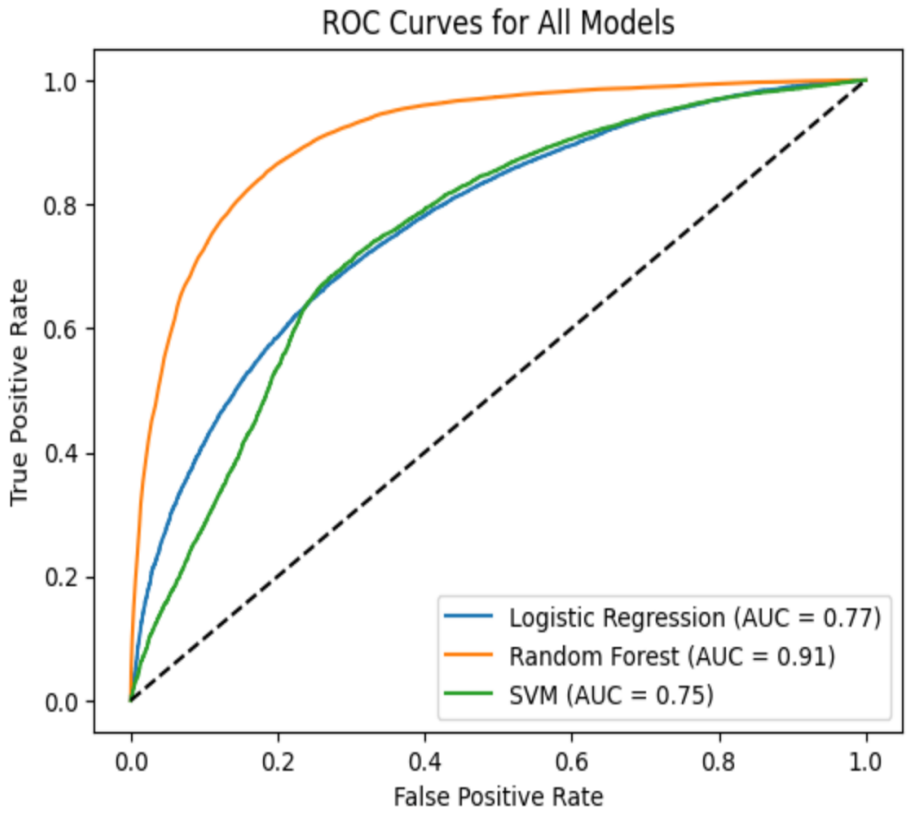


Fig 5. Model Performance Comparisons

- **ROC Curve**



```python
# ROC Curve
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curves for All Models')
plt.legend()
plt.show()
```

```
== Logistic Regression ==
              precision    recall  f1-score   support

           0       0.65      0.43      0.52     10790
           1       0.75      0.88      0.81     21290

    accuracy                           0.73     32080
   macro avg       0.70      0.66      0.66     32080
weighted avg       0.72      0.73      0.71     32080


== Random Forest ==
              precision    recall  f1-score   support

           0       0.81      0.74      0.77     10790
           1       0.87      0.91      0.89     21290

    accuracy                           0.85     32080
   macro avg       0.84      0.82      0.83     32080
weighted avg       0.85      0.85      0.85     32080


== SVM ==
              precision    recall  f1-score   support

           0       0.68      0.40      0.50     10790
           1       0.75      0.91      0.82     21290

    accuracy                           0.74     32080
   macro avg       0.72      0.65      0.66     32080
weighted avg       0.73      0.74      0.71     32080
```

- **Random Forest** has the highest AUC and dominates the others throughout most of the curve, indicating it's the best model in terms of distinguishing between the classes.

- **Logistic Regression** performs slightly better than **SVM** in this case.

- The closer the curve follows the left-hand border and then the top border of the ROC space, the better the model.

8 GitHub Repository and Collaboration

Repository: cap5771sp25-project

• Contents:

– Reports: Milestone1.pd, MileStone2.pdf

– Notebooks: Preprocessing and EDA

– Scripts: Python data handling scripts

– README.md: Reproduction instructions
• Collaborators: TA Jimmy (@JimmyRaoUF), Grader Daniyal (@abbasidaniyal), Dr. Cruz (@lcruz-cas), Dr. Grant (@cegme)

9 Conclusion and Next Steps

This milestone achieved the integration of multiple datasets, execution of advanced EDA, strategic feature engineering, and initial model training with comparative evaluations. The next milestone will include:

- Deployment of the best-performing model
- Fine-tuning with hyperparameter optimization
- Integration with a real-time conversational agent
- Extended evaluation using cross-validation and A/B testing

This lays the foundation for a robust and scalable AI-driven medication recommendation framework.