

Predictive Drug Classification:
Leveraging Patient Data for Personalized Treatment

NAME: MRITYUNJAY SHARMA

ROLL NO: 210107054

COURSE: CL653 AI&ML

DATE: 14/04/2024

Project Overview:

❖ Introduction:

Drug repurposing, also known as **drug repositioning**, is a critical pursuit in pharmacology, involving the utilization of existing medications for treating conditions not initially intended for. It offers a **cost-effective and expedited approach** compared to new drug development, simplifying clinical studies. Despite its potential, identifying novel applications for existing drugs is challenging due to the vast array of medications available and the **time-consuming** nature of screening each for new uses.

Furthermore, the discovery of both beneficial and adverse side effects is crucial for successful repurposing efforts, necessitating a comprehensive understanding of medication profiles.

❖ Objective:

The objective of this research paper is to propose a drug classification methodology aimed at **categorizing medications based on their applications**. By organizing drugs according to their intended usage, this approach aims to facilitate the process of drug repurposing, **streamlining the identification** of potential new therapeutic indications for existing medications.

Description of Project:

❖ Theoretical Background:

- Project lies in *pharmacovigilance*, focusing on detecting, assessing, understanding, and preventing adverse effects related to drugs.
- Growing interest in using social networks and online platforms to identify adverse drug reactions (ADRs).
- User-generated content on these platforms offers valuable insights into real-world patient experiences with medications.

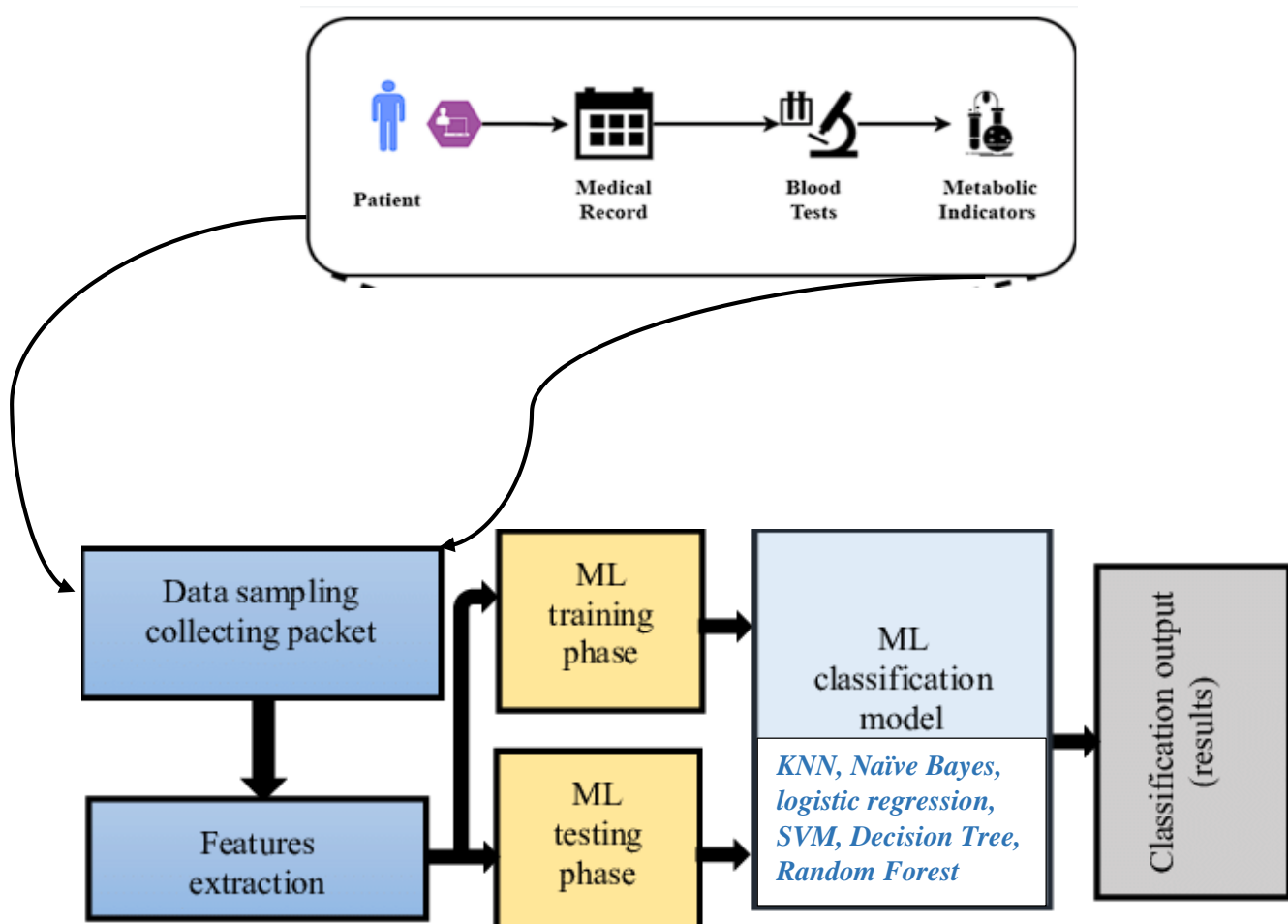
❖ Problem Statement:

- Challenge in efficiently and accurately extracting relevant information from unstructured social media text.
- Existing methods vary in effectiveness and efficiency, with dictionary-based approaches being common but potentially lacking in precision and recall.
- Lack of comprehensive resources and tools for ADR detection in languages other than English, such as Russian.

❖ Significance of Addressing this Issue:

- Early warning of potential drug safety issues through ADR detection from social media data.
- Complement to traditional *pharmacovigilance* methods, enhancing efficiency and cost-effectiveness.
- Addressing gap in ADR detection in non-English languages promotes more inclusive and comprehensive *pharmacovigilance* efforts.
- Potential to improve drug safety monitoring and enhance patient care through robust methods and resources for ADR detection.

Flowchart of Process & Model Implementation:



Data Sources and Data Description:

The dataset is taken from Kaggle.com which is a Drug Classification dataset. The target feature is Drug type. The feature sets are: *Age, Sex, Blood Pressure Levels (BP), Cholesterol Levels, Na to K Ratio*.

Data Preprocessing:

1. **Data Cleaning:** Handling of missing data points, which may involve imputation techniques such as mean substitution or deletion of records with missing values.

```
✓ [769] df.isnull().sum()  
0s  
  
Age          0  
Sex          0  
BP          0  
Cholesterol  0  
Na_to_K      0  
Drug         0  
dtype: int64
```


NO null values to be found

```
def remove_outliers_zscore(column):  
    z_scores = stats.zscore(column)  
    outlier_indices = abs(z_scores) > 3  
    column[outlier_indices] = column.mean()  
    return column
```

```
cleaned_column = remove_outliers_zscore(df['Na_to_K'])  
df['Na_to_K_cleaned'] = cleaned_column
```

2. **Normalization (numerical features):** Scaling numerical features to a consistent range to prevent biases in model training, using techniques like min-max scaling or z-score normalization.


3. Text Preprocess: One-hot encoding or Label encoding for converting categorical features to numerical features.

 X.head()



	Sex	BP	Cholesterol	Age_binned	Na_to_K_binned
0	F	HIGH	HIGH	21-30	20-30
1	M	LOW	HIGH	41-50	10-20
2	M	LOW	HIGH	41-50	10-20
3	F	NORMAL	HIGH	21-30	<10
4	F	LOW	HIGH	61-70	10-20

```
x = pd.get_dummies(X).astype(int)
```

 x.head()



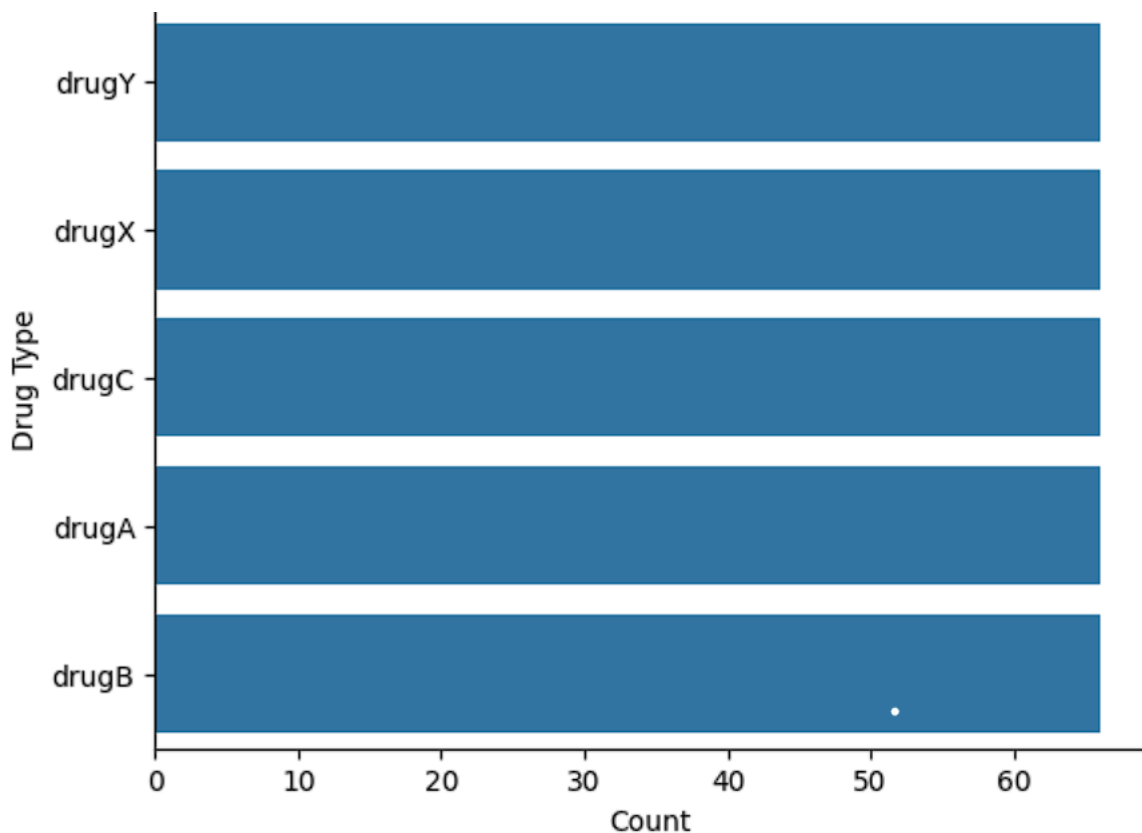
	Sex_F	Sex_M	BP_HIGH	BP_LOW	BP_NORMAL	Cholesterol_HIGH	Cholesterol_NORMAL	Age_binned_1-20	Age_binned_21-30	Age_binned_31-40	Age_binned_41-50	Age_binned_51-60	Age_binned_61-70	Age_binned_71-80	Na_to_K_binned_10-20	Na_to_K_binned_20-30	Na_to_K_binned_30-40	Na_to_K_binned_<10
0	1	0	1	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0
1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0
2	0	1	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0
3	1	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	1
4	1	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0

4. **Handling Imbalanced Data:** Apply techniques like oversampling or undersampling to balance class distributions if significant imbalances are observed.

✓ SMOTE

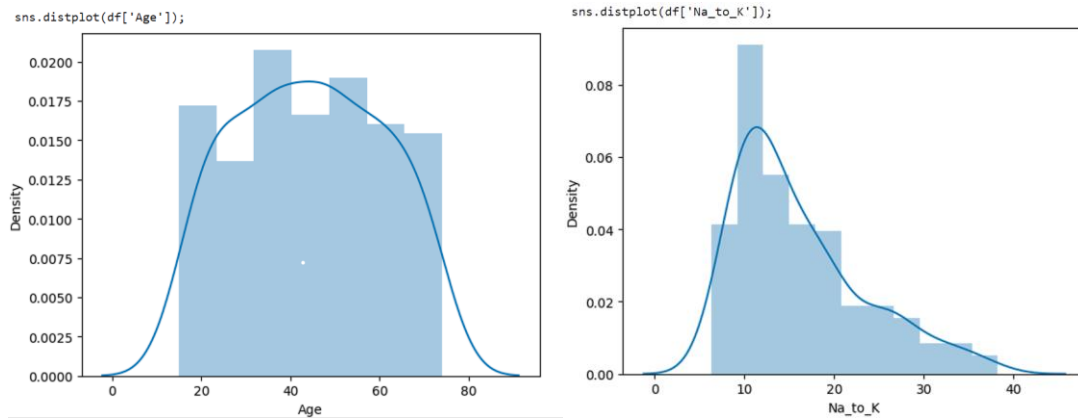
Since the number of 'DrugY' is more than other types of drugs, **oversampling is carried out to avoid overfitting.**

```
[800] X_train, y_train = SMOTE().fit_resample(X_train, y_train)
```



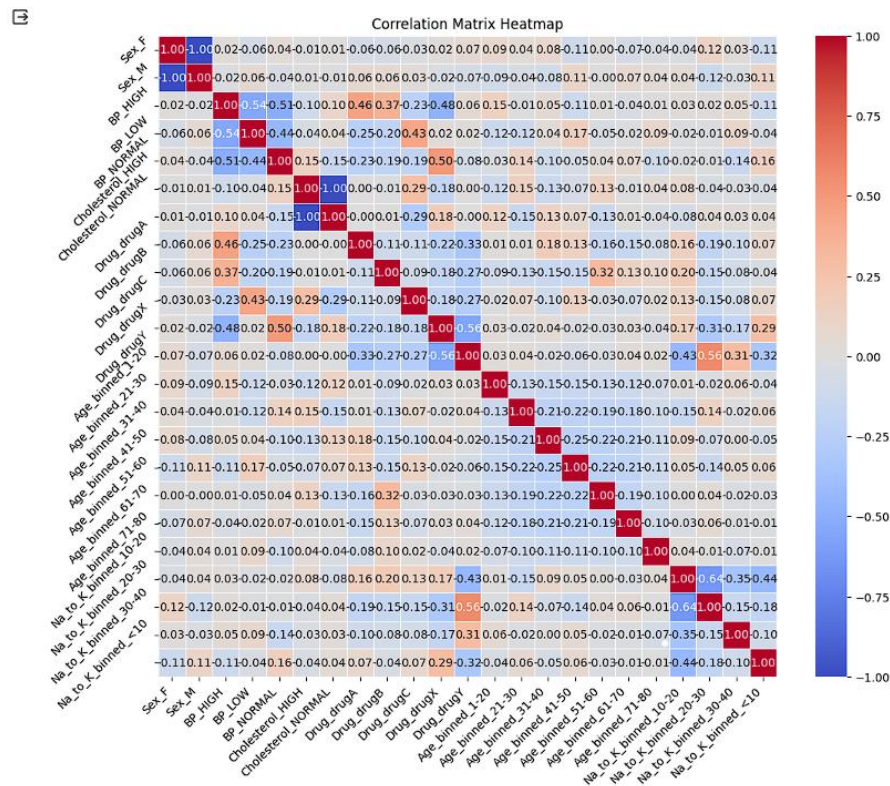
5. **Feature Engineering:** Explore potential interactions or combinations of features that may provide additional predictive power.

- Exploratory Data Analysis (EDA):** Conduct exploratory data analysis to gain insights into the relationships between features and the target variable. Visualize distributions, correlations, and patterns in the data using techniques like histograms, scatter plots, or correlation matrices. Identify any potential trends or outliers that may impact model training and interpretation.



The distribution of 'Age' column is **symetric**, since the skewness value between -0.5 and 0.5

The distribution of 'Na_to_K' column is **moderately skewed**, since the skewness value is **between 0.5 and 1**. It can also be seen from the histogram for 'Na_to_K' column



Strategies for AI/ML Model Development

- **Model Development:**

Considering the drug classification problem with features such as age, sex, blood pressure levels, cholesterol levels, and Na to Potassium ratio, several machine learning models could be suitable :- Logistic Regression, K Nearest Neighbors (KNN), Naïve Bayes, Support Vector Machine (SVM), Decision Trees, Random Forest.

The rationale for choosing these models lies in their *versatility*, *interpretability*, and ability to handle both numerical and categorical features present in the dataset. Ensemble methods like Random Forest is particularly well-suited for handling complex interactions between features.

- **Training:**

1. **Splitting data:** Divide the dataset into training and testing sets, typically using a 70-30 or 80-20 split ratio.

```
X = df.drop(['Drug'], axis=1)
y = df['Drug']
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, random_state = 54)
```

2. **Model training:** Fit the selected machine learning models (Logistic Regression, Random Forest, SVM) on the training data using appropriate libraries such as scikit-learn in Python.

```
model7 = RandomForestClassifier(max_leaf_nodes=5)
model7.fit(X_train, y_train)

y_pred7 = model7.predict(X_test)

print(classification_report(y_test, y_pred7))
print(confusion_matrix(y_test, y_pred7))

acc7 = accuracy_score(y_pred7, y_test)
print('Random Forest accuracy is: {:.2f}%'.format(acc7*100))
```

3. **Hyperparameter tuning:** Optimize model hyperparameters using techniques like grid search or random search to improve model performance.

```
def hyperparameter_tuning(X, y, models, scoring, cv=5, plot_results=True):
    best_models = {}
    for model_name, model in models.items():
        print(f"Tuning {model_name}...")
        X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, random_state=54)

        param_grid = define_param_grid(model)

        grid_search = GridSearchCV(model, param_grid, scoring=scoring, cv=cv, n_jobs=-1)
        grid_search.fit(X_train, y_train)

        best_model = grid_search.best_estimator_
        best_score = grid_search.best_score_
        best_models[model_name] = (best_model, best_score)

    return best_models

def define_param_grid(model):
    param_grids = {
        LogisticRegression: {'max_iter': [100, 1000, 5000], 'solver': ['liblinear']},
        CategoricalNB: {},
        GaussianNB: {},
        RandomForestClassifier: {'n_estimators': [100, 500, 1000], 'max_leaf_nodes': [5, 10, 20, 30]},
        DecisionTreeClassifier: {'max_leaf_nodes': [5, 10, 15, 20]},
        SVC: {'kernel': ['linear', 'poly', 'rbf'], 'max_iter': [250, 300, 350]},
        KNeighborsClassifier: {'n_neighbors': [10, 15, 20, 25]}
    }

    return param_grids.get(type(model), {})
```

```
models = {
    'Logistic Regression': LogisticRegression(),
    'K Neighbors': KNeighborsClassifier(),
    'SVM': SVC(),
    'Categorical NB': CategoricalNB(),
    'Gaussian NB': GaussianNB(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
}

# Perform hyperparameter tuning
x = pd.get_dummies(X).astype(int)
best_models = hyperparameter_tuning(x, y, models, scoring='accuracy')
```

```
Tuning Logistic Regression...
Tuning K Neighbors...
Tuning SVM...
Tuning Categorical NB...
Tuning Gaussian NB...
Tuning Decision Tree...
Tuning Random Forest...
```

4. **Cross-validation:** Perform k-fold cross-validation on the training data to assess model generalization and stability.

- **Evaluation and Validation:**

- **Evaluation Metrics:** For evaluating the model use accuracy, F1-score and confusion matrix. along with them we can use ROC-AUC curve to distinguish classes across different thresholds.

	precision	recall	f1-score	support
drugA	0.50	0.44	0.47	9
drugB	0.47	0.88	0.61	8
drugC	0.62	1.00	0.77	5
drugX	0.81	1.00	0.90	13
drugY	1.00	0.52	0.68	25
accuracy			0.70	60
macro avg	0.68	0.77	0.69	60
weighted avg	0.78	0.70	0.70	60

```
[[ 4  5  0  0  0]
 [ 1  7  0  0  0]
 [ 0  0  5  0  0]
 [ 0  0  0 13  0]
 [ 3  3  3  3 13]]
```

Random Forest accuracy is: 70.00%

Metric: **Accuracy**

- **Validation Strategy:** Implement k-fold cross-validation (e.g., 5-fold or 10-fold) to ensure robustness and generalizability of the model.

Results:

	Model	before_tuning	after_tuning
0	Logistic Regression	80.000000	74.285714
1	K Neighbors	56.666667	72.142857
2	SVM	75.000000	76.428571
3	Categorical NB	70.000000	78.571429
4	Gaussian NB	76.666667	79.285714
5	Decision Tree	45.000000	79.285714
6	Random Forest	65.000000	80.000000

