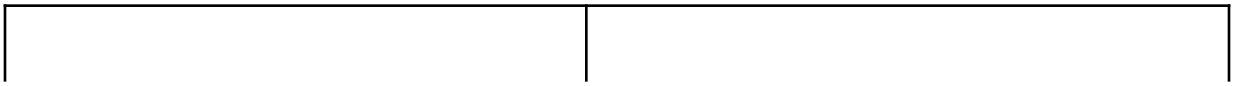

Comparative Analysis of Random Forest for Diabetes Classification: Impact of Hyperparameter Tuning and Feature Selection via Pearson and Spearman Correlation

Kent Christopher Hansel	**2602067634**
Antony Laurence Cornelis	**2602067773**
Gregorius Geraldin	**2440043982**
Kevin Purnomo	**2602059645**

BINUS
UNIVERSITY

2024

Table of Content

I. Introduction

Background:

Diabetes is a long-term metabolic disease marked by elevated blood sugar levels. If unchecked, diabetes can cause serious side effects such kidney failure, heart disease, and blindness. It arises when your body isn't reacting to the effects of insulin appropriately or when your pancreas produces too little or no insulin at all. The World Health Organization (WHO) estimates that 422 million people worldwide have diabetes in 2014; this figure is projected to increase dramatically in the upcoming years.

Predicting diabetes early on can be very important for reducing the negative health impacts that come with the condition. Healthcare professionals can identify people who are at a high risk of getting diabetes with the help of accurate prediction models. This allows for early intervention and lifestyle modifications that can delay or prevent the beginning of the illness. Predictive models can also assist healthcare organizations allocate resources more efficiently by focusing preventive treatment on the most vulnerable. Within this framework, data mining techniques have become highly effective instruments for gleaning insightful information from big datasets and creating reliable predictive models.

Through the extraction of insightful information from complex datasets, data mining improves the prediction of diabetes by shedding light on the significant contributions of genetic, lifestyle, and environmental factors to diabetes risk. This understanding facilitates the development of effective preventive and intervention programs that may not be fully captured by traditional methods.

Random Forest is a robust learning method that combines multiple decision trees to improve classification accuracy and reduce overfitting, making it well-suited for diabetes prediction. To enhance model performance, feature selection using Pearson and Spearman correlation coefficients identifies the most relevant variables by measuring linear and monotonic relationships, respectively. Additionally, hyperparameter tuning, which optimizes parameters such as the number of trees and maximum depth, is critical for maximizing model accuracy and generalization. Numerous studies, including those by Smith et al. (2019), Zhang et al. (2020), and Lee and Kim (2021), have demonstrated the effectiveness of Random Forest, correlation analysis, and hyperparameter tuning in diabetes prediction.

Problem Definition:

- What are the main factors contributing to diabetes risk prediction?
- How can feature selection and hyperparameter tuning optimize Random Forest performance for diabetes prediction?
- How do Pearson and Spearman correlation and hyperparameter tuning affect Random Forest accuracy in predicting diabetes?

Solution:

- To develop a predictive model that can accurately identify individuals at high risk of developing diabetes.
- Utilize data mining techniques to analyze large datasets and identify primary factors contributing to diabetes onset.
- Comparing 5 models of the same classification algorithm to ensure best feature selection and best parameter tuning.

Other Study:

The study by Rastogi Rashi and Bansal Mamta titled "Diabetes Prediction Model Using Data Mining Techniques" proposes a diabetes prediction model employing Random Forest, Support Vector Machine (SVM), Logistic Regression, and Naive Bayes. The model, implemented in Python and evaluated with a Kaggle dataset, demonstrates that logistic regression achieved the highest accuracy of 82.46% among the techniques tested.

Devi et al. (2022) assessed the potency of algorithms including Random Forest, Support Vector Machine (SVM), Logistic Regression, K-Nearest Neighbors (KNN) Classifier, and the Logistic algorithm. The performance of these algorithms was thoroughly compared in the study using their accuracy, precision, recall, and F1 score as metrics. The scientists discovered that Random Forest outperformed the other assessed approaches in terms of prediction, especially when it came to accuracy and resilience. The study demonstrated how Random Forest is particularly useful for diabetes prediction because of its capacity to manage huge datasets with intricate feature relationships. This study highlights Random Forest's potential as a trustworthy tool for diabetes management's early diagnosis and intervention.

II. Data and Preprocessing

2.1 Data Description

The dataset used is gathered from an open dataset repository Kaggle containing a BRFSS csv file. The Behavioral Risk Factor Surveillance System (BRFSS) is a health-related telephone survey that is collected annually by the CDC. This dataset is a clean dataset of 253,680 survey responses to the CDC's BRFSS2015. The target variable Diabetes_binary has 2 classes, 0 is for no diabetes, and 1 is for prediabetes or diabetes. This dataset has 21 feature variables with 1 targeted output and is not balanced.

Table 1. Data Used

Feature Name	**Data Type**	**Description**
Diabetes_binary	Float (binary)	Targeted output of 1 or 0
HighBP	Float (binary)	Have high blood pressure
HighChol	Float (binary)	Have high cholesterol
CholCheck	Float (binary)	Cholesterol check
BMI	Float	Body Mass Index
Smoker	Float (binary)	Is a smoker
Stroke	Float (binary)	Ever had a stroke
HeartDiseaseorAttack	Float (binary)	Ever had a heart attack
PhysActivity	Float (binary)	Physically active or not
Fruits	Float (binary)	Eat fruits or not
Veggies	Float (binary)	Eat vegetables or not
HvyAlcoholConsump	Float (binary)	Drink alcohol or not
AnyHealthCare	Float (binary)	Have health care
NoDocbcCost	Float (binary)	Need a doctor
GenHlth	Float	General health scale (1-5)
MentHlth	Float	Days of poor mental health (1-31)

PhysHlth	Float	Physical injury in last 30 days
DiffWalk	Float (binary)	Difficulty walking
Sex	Float (binary)	Male or Female
Age	Float	13 categories of age

2.2 Data Preprocessing

The data is preprocessed in 4 main steps which are:

- Data Cleaning
- Data Transformation
- Feature Selection
- Data Splitting: Dividing the dataset into training, validation, and test sets.

2.2.1 Data Cleaning

Duplicate rows have been removed from the dataset, ensuring data integrity. Given that many features are categorical binary floats (0 or 1), detecting outliers becomes challenging. However, employing the Interquartile Range (IQR) method revealed the data to be mostly outlier-free. However, the "BMI" feature has some outliers, which is about 9847 data. We only checked the "BMI" feature because it is the only feature that contains continuous data and most of the other features are categorical data which are already encoded in the dataset.

```python
def check_outlier(df):
    Q1 = df.quantile(0.25)
    Q3 = df.quantile(0.75)
    IQR = Q3 - Q1

    outliers = df[(df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))]
    return outliers


def print_outlier_BMI(df):
    outliers = check_outlier(df['BMI'])
    print(f'Number of Outliers for BMI: {len(outliers)}')

print_outlier_BMI(df)
```

```python
def handle_outlier_BMI(df):
    Q1 = df['BMI'].quantile(0.25)
    Q3 = df['BMI'].quantile(0.75)
    IQR = Q3-Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Drop the outliers
    lower_outliers = np.where(df['BMI'] <= lower_bound)[0]
    upper_outliers = np.where(df['BMI'] >= upper_bound)[0]
    df.drop(index = upper_outliers, inplace = True)
    df.drop(index = lower_outliers, inplace = True)
    df.reset_index(drop=True, inplace=True)

handle_outlier_BMI(df)
```

Figure 1. (a) Display the number of outliers; (b) Outlier handling

2.2.2 Data Transformation

We use standard scaler (Standardization) to avoid features with higher values from dominating the objective function and decrease the performance of the model. The performance of a Random Forest model can be enhanced by applying the Standard Scaler to the input data prior to training, particularly in cases where the features have significantly different scales or

units. It is a straightforward but powerful preprocessing technique that can improve the model's capacity to extract useful patterns from the data and produce predictions that are more accurate.

2.2.3 Feature Selection

Spearman's and Pearson's correlation analysis algorithms are used to select important features from the dataset. As a result, two different datasets are created by removing the top 5 least correlated features according to each correlation method. This approach aims to diminish data dimensionality and computational complexity, thus conserving computing resources and time. Five features were picked because of the similarity of pearson and spearman correlation analysis result giving the same output almost in every feature.

```
# Drop Features that have low correlation with Diabetes
# Drop based on pearson correlation
columns_to_drop_pearson = ["Fruits" , "Veggies" , "Sex" , "NoDocbcCost" , "AnyHealthcare"]
columns_to_drop_spearman = ["Fruits" , "MentHlth" , "Sex" , "NoDocbcCost" , "AnyHealthcare"]

df_pearson = df.drop(columns_to_drop_pearson, axis = 1)
df_spearman = df.drop(columns_to_drop_spearman, axis = 1)
```

Figure 2. Feature selection code

2.2.4 Data Splitting

The dataset is initially divided into training and testing sets with an 80% training data to 20% testing data ratio..

```
from sklearn.model_selection import train_test_split
X_train_p , X_test_p , y_train_p , y_test_p = train_test_split(X_pearson, y_pearson, test_size=0.2 , random_state=42)
```

Figure 3. Data splitting with test train

III. Exploratory Data Analysis (Dataset Problem Solving)

The data has a total of 22 columns (feature variables + targeted output) and 253.680 instances of data with no missing values.

```
Data columns (total 22 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   Diabetes_binary       253680 non-null  float64
 1   HighBP                253680 non-null  float64
 2   HighChol              253680 non-null  float64
 3   CholCheck             253680 non-null  float64
 4   BMI                   253680 non-null  float64
 5   Smoker                253680 non-null  float64
 6   Stroke                253680 non-null  float64
 7   HeartDiseaseorAttack  253680 non-null  float64
 8   PhysActivity          253680 non-null  float64
 9   Fruits                253680 non-null  float64
 10  Veggies               253680 non-null  float64
 11  HvyAlcoholConsump     253680 non-null  float64
 12  AnyHealthcare         253680 non-null  float64
 13  NoDocbcCost           253680 non-null  float64
 14  GenHlth               253680 non-null  float64
 15  MentHlth              253680 non-null  float64
 16  PhysHlth              253680 non-null  float64
 17  DiffWalk              253680 non-null  float64
 18  Sex                   253680 non-null  float64
 19  Age                   253680 non-null  float64
 20  Education             253680 non-null  float64
 21  Income                253680 non-null  float64
dtypes: float64(22)
```

Figure 4. Data used after checking

```
1 df.isna().sum()
✓ 0.0s

Diabetes_binary         0
HighBP                  0
HighChol                0
CholCheck               0
BMI                     0
Smoker                  0
Stroke                  0
HeartDiseaseorAttack    0
PhysActivity            0
Fruits                  0
Veggies                 0
HvyAlcoholConsump       0
AnyHealthcare           0
NoDocbcCost             0
GenHlth                 0
MentHlth                0
PhysHlth                0
DiffWalk                0
Sex                     0
Age                     0
Education               0
Income                  0
dtype: int64
```

Figure 5. Data with missing values

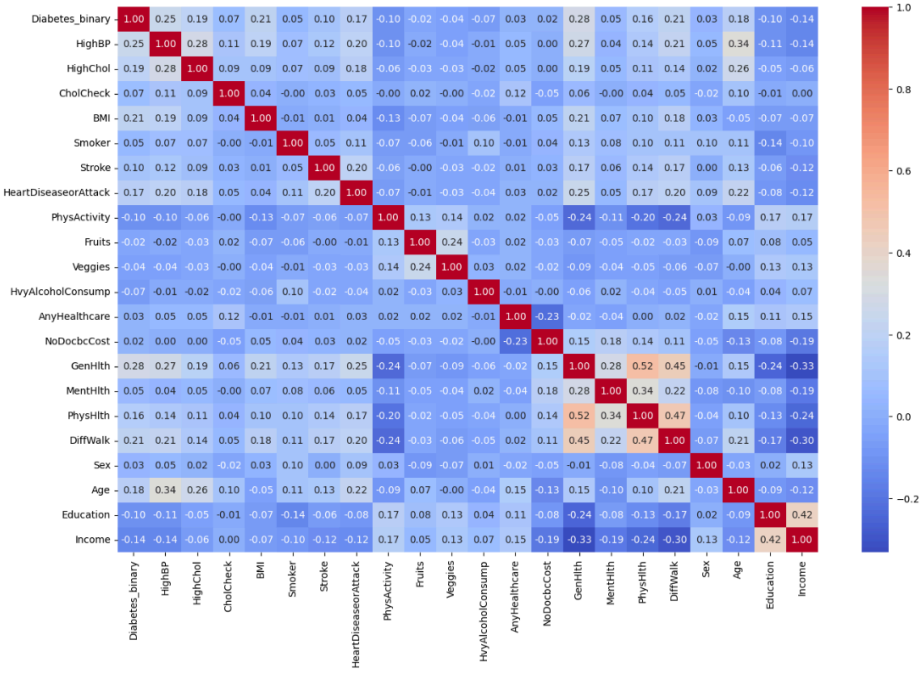

Figure 6. Correlation Matrix result (pearson)

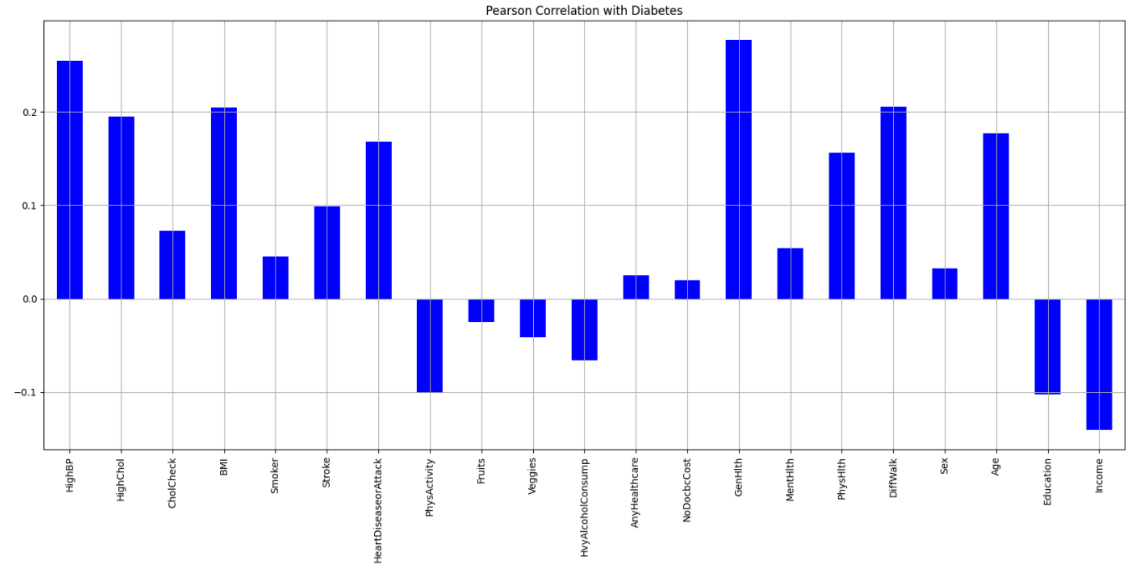

Figure 7. Pearson Correlation Analysis

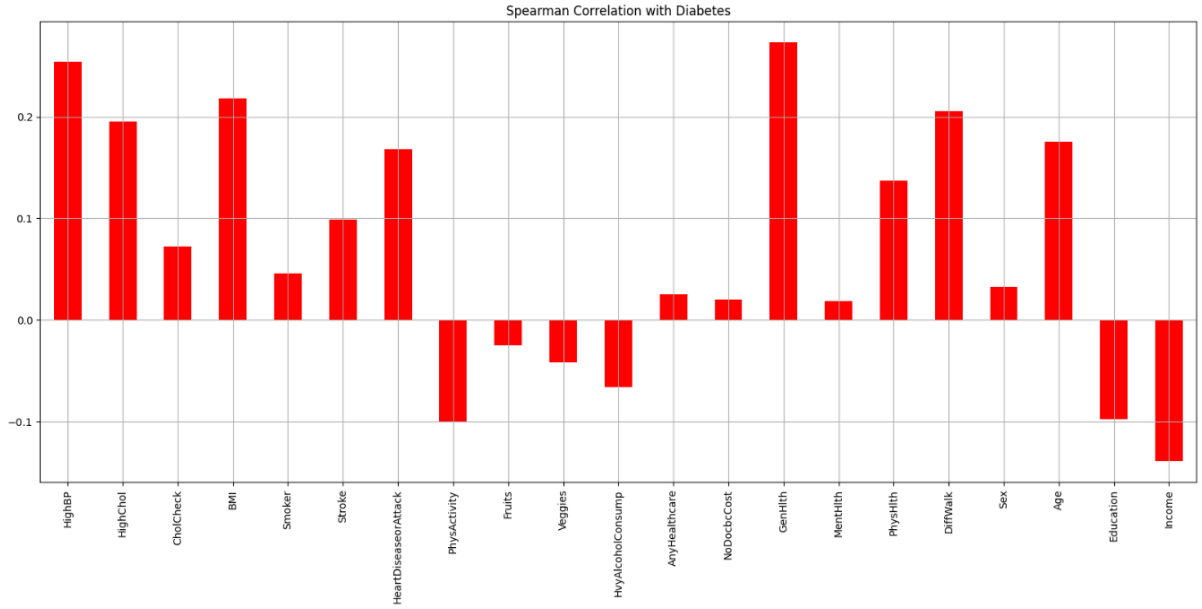

Figure 8. Spearman Correlation Analysis

Diabetes_binary	1.000000
GenHlth	0.265285
HighBP	0.246928
BMI	0.202706
HighChol	0.189925
DiffWalk	0.187311
Age	0.181838
HeartDiseaseorAttack	0.167169
PhysHlth	0.143015
Income	0.132002
Stroke	0.099215
Education	0.098386
PhysActivity	0.090788
CholCheck	0.071039
HvyAlcoholConsump	0.064051
Smoker	0.046208
MentHlth	0.044812
Veggies	0.040681
Sex	0.040387
AnyHealthcare	0.026252
Fruits	0.022167
NoDocbcCost	0.016332

Figure 9. Pearson Correlation Result

Diabetes_binary	1.000000
GenHlth	0.261647
HighBP	0.246928
BMI	0.199843
HighChol	0.189925
DiffWalk	0.187311
Age	0.180803
HeartDiseaseorAttack	0.167169
Income	0.130552
PhysHlth	0.123360
Stroke	0.099215
Education	0.092840
PhysActivity	0.090788
CholCheck	0.071039
HvyAlcoholConsump	0.064051
Smoker	0.046208
Veggies	0.040681
Sex	0.040387
AnyHealthcare	0.026252
Fruits	0.022167
NoDocbcCost	0.016332
MentHlth	0.009484

Figure 10. Spearman Correlation Result

Based on the figures and correlation scores presented earlier, both Pearson and Spearman correlation analyses yield nearly identical correlation scores. Consequently, the top 5 least correlated features were removed from each dataset: "Fruits," "Veggies," "Sex," "NoDocbcCost," and "AnyHealthcare" in the Pearson correlation dataset, and "Fruits," "MentHlth," "Sex," "NoDocbcCost," and "AnyHealthcare" in the Spearman correlation dataset. Resulting with 2 datasets to be used to train and test the model.

IV. Method

Every step and process is performed using Python NoteBook and libraries such as pandas, numpy, sklearn, and imblearn, to simplify and streamline the workflow.

1. **Pearson Correlation**

   Pearson correlation is a statistical measure that evaluates the linear relationship between two continuous variables. It ranges from -1 to 1, where 1 indicates a perfect positive linear relationship, -1 indicates a perfect negative linear relationship, and 0 indicates no linear relationship.

```
correlation_pearson = df.corr(method='pearson')
correlation_pearson['Diabetes_binary'].abs().sort_values(ascending = False)
```

Figure 11 Pearson Correlation code

2. **Spearman Correlation**

   Spearman correlation assesses the strength and direction of a monotonic relationship between two variables, regardless of whether the relationship is linear. It also ranges from -1 to 1, similar to Pearson correlation.

```
correlation_spearman = df.corr(method='spearman')
correlation_spearman['Diabetes_binary'].abs().sort_values(ascending = False)
```

Figure 12 Spearman Correlation Code

3. **Hyperparameter Tuning**

   Hyperparameter tuning involves adjusting the parameters of a machine learning model to optimize its performance. Tuning these parameters, which are not learned from the data but set before the training process, can significantly influence the model's accuracy, optimizes the model's performance, reducing overfitting, or enhancing convergence speed, and generalization ability.

```
rf = RandomForestClassifier(random_state=42)
params = {
        'n_estimators': [10, 30, 100, 300, 400],
        'max_depth': [3, 5, 7, 8, 9, 10],
        'min_samples_split': [2, 4, 6, 8, 10],
        'max_features': ['sqrt', 'log2'],
        'bootstrap': [True]
    }

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
# Instantiate cross validated Decision Tree random search
rs_rf_spearman= RandomizedSearchCV(rf, params, cv=skf, n_jobs=-1, verbose=1, random_state=42)

# fit knn to training data
rs_rf_spearman.fit(X_train_s, y_train_s)

rf_tuned_spearman = rs_rf_spearman.best_estimator_
rf_tuned_spearman.fit(X_train_s, y_train_s)

y_pred_tuned_spearman = rf_tuned_spearman.predict(X_test_s)
```

Figure 13 Hyperparameter Tuning Code

4. **Random Forest**

Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees. It is highly effective in handling large datasets with numerous features and complex interactions.

Combining Pearson and Spearman correlation analyses with Random Forest provides a robust feature selection and model training approach:

a. **Feature Selection**: Pearson identifies features with linear relationships, while Spearman captures monotonic (non-linear) relationships.
b. **Model Training**: Random Forest uses these features to handle both linear and non-linear interactions, enhancing prediction accuracy.
c. **Dataset Relevance**: The dataset includes features with both linear and non-linear relationships to diabetes. Using both correlation methods ensures all significant relationships are considered, leading to a more accurate Random Forest model.

The purpose of this study is to identify features in the dataset that have strong relationships with the target variable using both Pearson and Spearman correlation analyses, and to build an accurate predictive model using Random Forest with hyperparameter tuning. Pearson correlation is used to calculate the linear relationship coefficient for each feature, while Spearman correlation assesses monotonic relationships, even if they are non-linear.

Features with high Pearson and Spearman correlation coefficients are considered important and prioritized in the feature selection process. The Random Forest model is then trained using these selected features, and hyperparameter tuning, including optimizing the number of trees and maximum depth, is performed to enhance model performance. The outcome is a robust predictive model capable of accurately classifying diabetes by leveraging the most relevant features identified through correlation analyses and optimized hyperparameters.

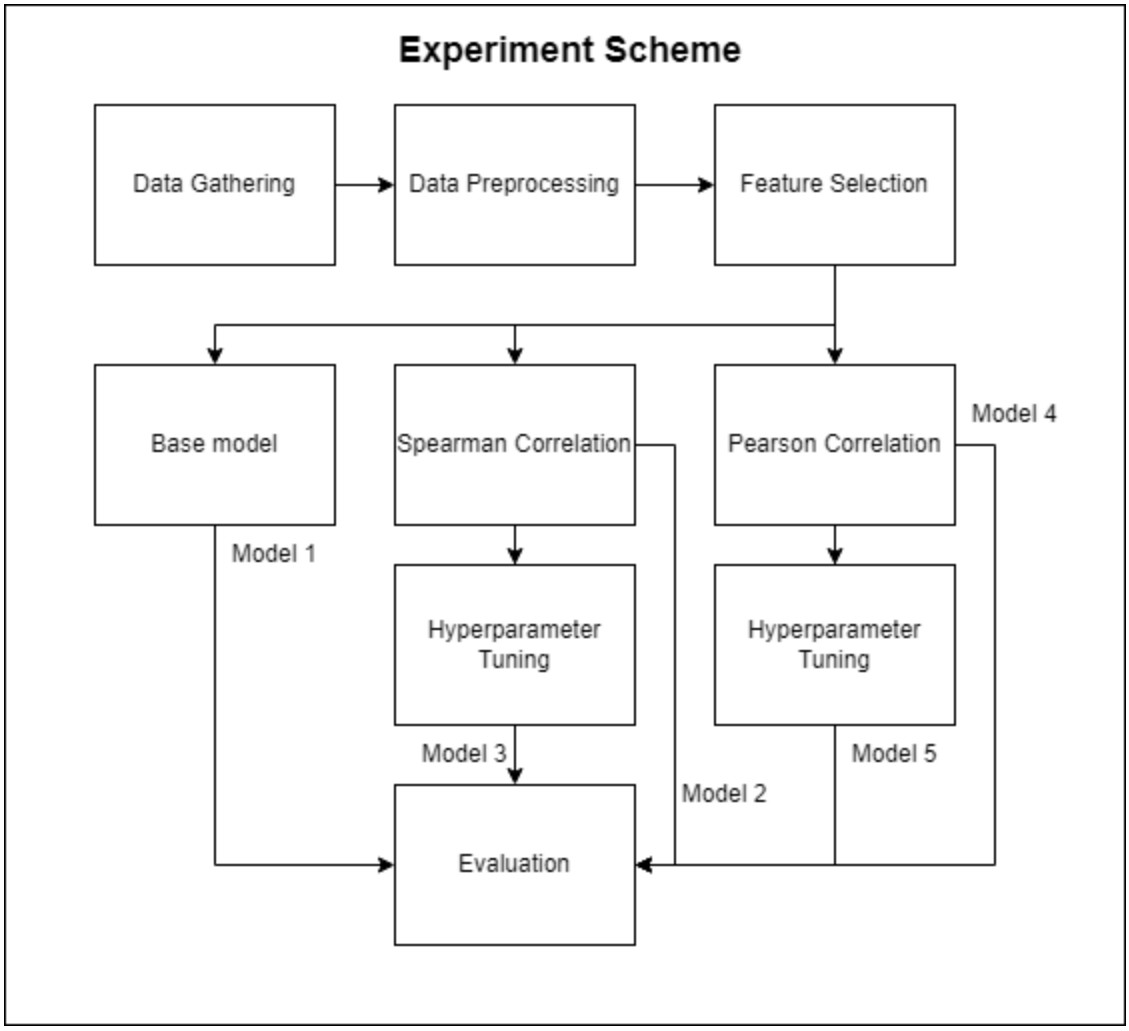

Figure 14. Experiment scheme

V. Result and Analysis

Unfortunately, due to the high dimensionality of our dataset, it is not feasible to plot the Random Forest classifier model. Below are shown the application of the Random Forest classifier model to our dataset to predict diabetes.

Applied base model of Random Forest Random Classification without feature selection and without hyperparameter tuning:

```
X = df.drop("Diabetes_binary", axis = 1)
y = df["Diabetes_binary"]

from sklearn.model_selection import train_test_split
X_train , X_test, y_train, y_test= train_test_split(X, y, test_size=0.2 , random_state=42)

rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)

print('Testing accuracy: ',accuracy_score(y_pred, y_test))
print('Training acacuracy:',rf.score(X_train, y_train))
print(classification_report(y_test, y_pred))
```

Figure 15. Random Forest without feature selection and without hyperparameter tuning

Applied model of Random Forest Classification with features from Pearson correlation analysis and with Hyperparameter Tuning in python:

```
# use random forest
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold
rf = RandomForestClassifier(random_state=42)
params = {
        'n_estimators': [10, 30, 100, 300, 400],
        'max_depth': [3, 5, 7, 8, 9, 10],
        'min_samples_split': [2, 4, 6, 8, 10],
        'max_features': ['sqrt', 'log2'],
        'bootstrap': [True]
    }

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
# Instantiate cross validated Decision Tree random search
rs_rf_pearson= RandomizedSearchCV(rf, params, cv=skf, n_jobs=-1, verbose=1, random_state=42)

# fit knn to training data
rs_rf_pearson.fit(X_train_p, y_train_p)

rf_tuned_pearson = rs_rf_pearson.best_estimator_
rf_tuned_pearson.fit(X_train_p, y_train_p)

y_pred_tuned_pearson = rf_tuned_pearson.predict(X_test_p)
```

Figure 16. Random Forest with feature Pearson correlation and hyperparameter tuning

Applied model of Random Forest Classification without features from Pearson correlation analysis and without Hyperparameter Tuning in python:

```
rf_vanilla_pearson = RandomForestClassifier(random_state=42)
rf_vanilla_pearson.fit(X_train_p, y_train_p)
y_pred_vanilla_pearson = rf_vanilla_pearson.predict(X_test_p)
```

Figure 17. Random Forest with feature pearson correlation and without hyperparameter tuning

Applied model of Random Forest Classification with features from Spearman correlation analysis and with Hyperparameter Tuning in python:

```
rf = RandomForestClassifier(random_state=42)
params = {
        'n_estimators': [10, 30, 100, 300, 400],
        'max_depth': [3, 5, 7, 8, 9, 10],
        'min_samples_split': [2, 4, 6, 8, 10],
        'max_features': ['sqrt', 'log2'],
        'bootstrap': [True]
    }

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
# Instantiate cross validated Decision Tree random search
rs_rf_spearman= RandomizedSearchCV(rf, params, cv=skf, n_jobs=-1, verbose=1, random_state=42)

# fit knn to training data
rs_rf_spearman.fit(X_train_s, y_train_s)

rf_tuned_spearman = rs_rf_spearman.best_estimator_
rf_tuned_spearman.fit(X_train_s, y_train_s)

y_pred_tuned_spearman = rf_tuned_spearman.predict(X_test_s)
```

Figure 18. Random Forest with feature spearman correlation and with hyperparameter tuning

Applied model of Random Forest Classification with features from Spearman correlation analysis and without Hyperparameter Tuning in python:

```
rf_vanilla_spearman = RandomForestClassifier(random_state=42)
rf_vanilla_spearman.fit(X_train_s, y_train_s)
y_pred_vanilla_spearman = rf_vanilla_spearman.predict(X_test_s)
```

Figure 19. Random Forest with feature spearman correlation and without hyperparameter tuning

Base model Random Forest Classification without feature selection and Hyperparameter Tuning:

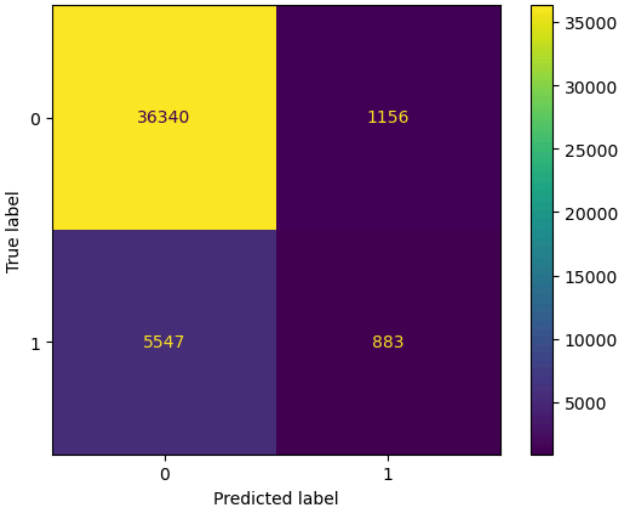

Figure 20. Random Forest without feature selection and hyperparameter tuning result

Random Forest Classification with features from Pearson correlation analysis and with Hyperparameter Tuning confusion matrix result:

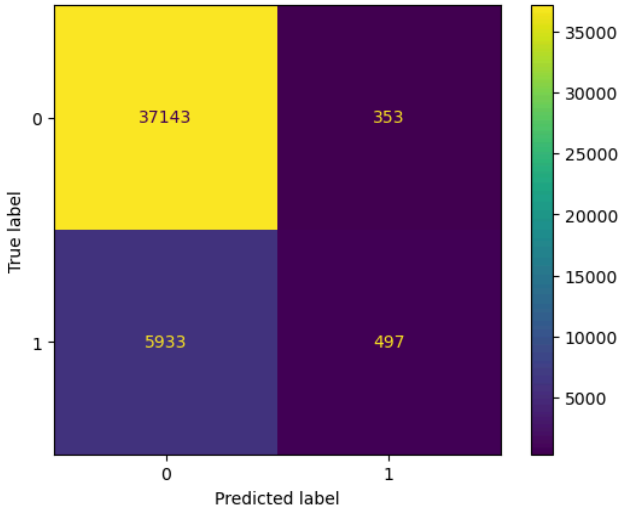

Figure 21. Random Forest with feature Pearson correlation and with hyperparameter tuning result

Random Forest Classification with features from Pearson correlation analysis and without Hyperparameter Tuning confusion matrix result:

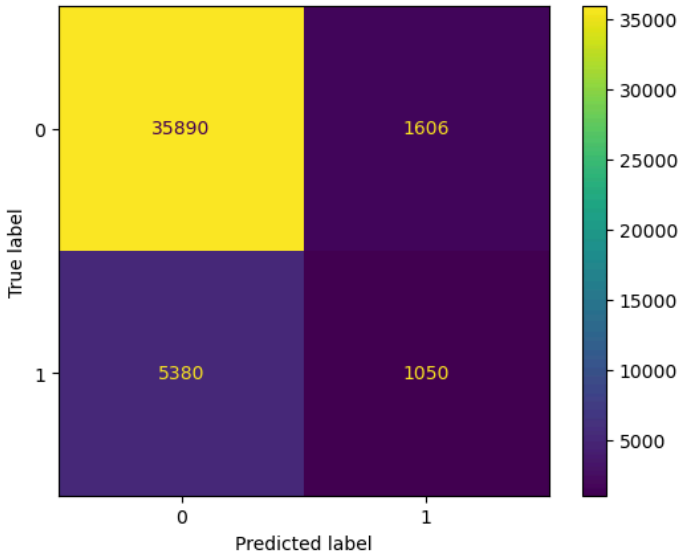

Figure 22. Random Forest with feature Pearson correlation and without hyperparameter tuning result

Random Forest Classification with features from Spearman correlation analysis and with Hyperparameter Tuning confusion matrix result:

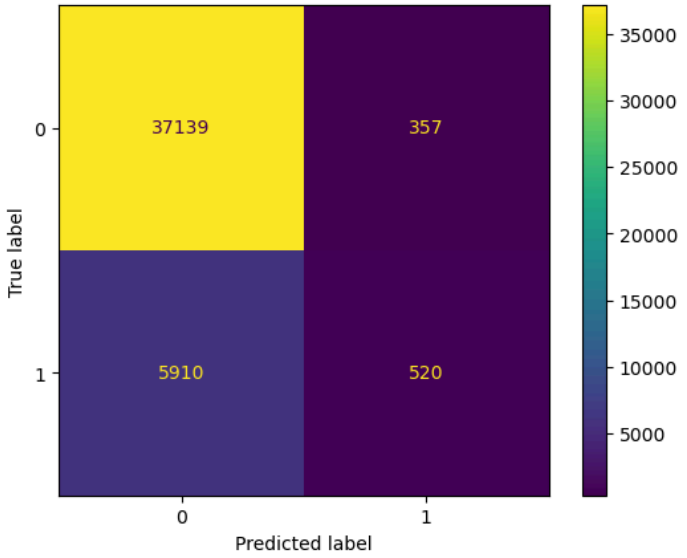

Figure 23. Random Forest with feature Spearman correlation and with hyperparameter tuning result

Random Forest Classification with features from Spearman correlation analysis and without Hyperparameter Tuning confusion matrix result:

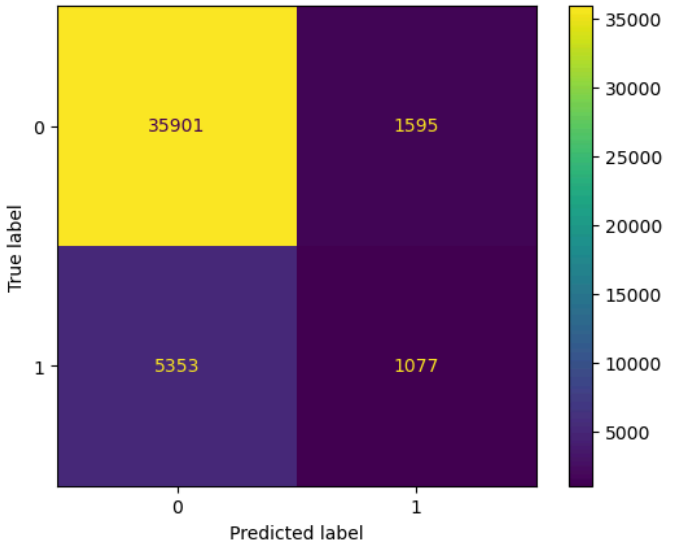

Figure 24. Random Forest with feature Spearman correlation and without hyperparameter tuning result

VI. Evaluation

Base model Random Forest Classification without feature selection and Hyperparameter Tuning accuracy:

```
Testing accuracy:  0.847402449574284
Training accuracy: 0.9943029185448254
Classification report:
              precision    recall  f1-score   support

           0       0.87      0.97      0.92     37496
           1       0.43      0.14      0.21      6430

    accuracy                           0.85     43926
   macro avg       0.65      0.55      0.56     43926
weighted avg       0.80      0.85      0.81     43926
```

Figure 25 Random Forest Classification without feature selection and with hyperparameter tuning accuracy

Random Forest Classification with features from Pearson correlation analysis and without Hyperparameter Tuning accuracy:

```
Testing accuracy:  0.84095979602058
Training accuracy: 0.9842234667395164
Classification report:
              precision    recall  f1-score   support

           0       0.87      0.96      0.91     37496
           1       0.40      0.16      0.23      6430

    accuracy                           0.84     43926
   macro avg       0.63      0.56      0.57     43926
weighted avg       0.80      0.84      0.81     43926
```

Figure 26. Random Forest Classification with feature from Pearson correlation analysis and without hyperparameter tuning accuracy

Random Forest Classification with features from Pearson correlation analysis and with Hyperparameter Tuning accuracy:

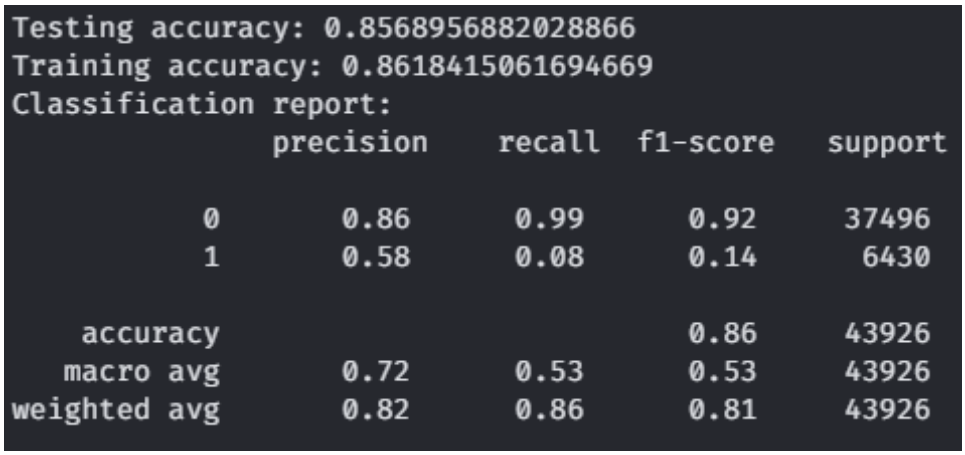

Figure 27. Random Forest Classification with feature from Pearson correlation analysis and with hyperparameter tuning accuracy

Random Forest Classification with features from Spearman correlation analysis and without Hyperparameter Tuning accuracy:

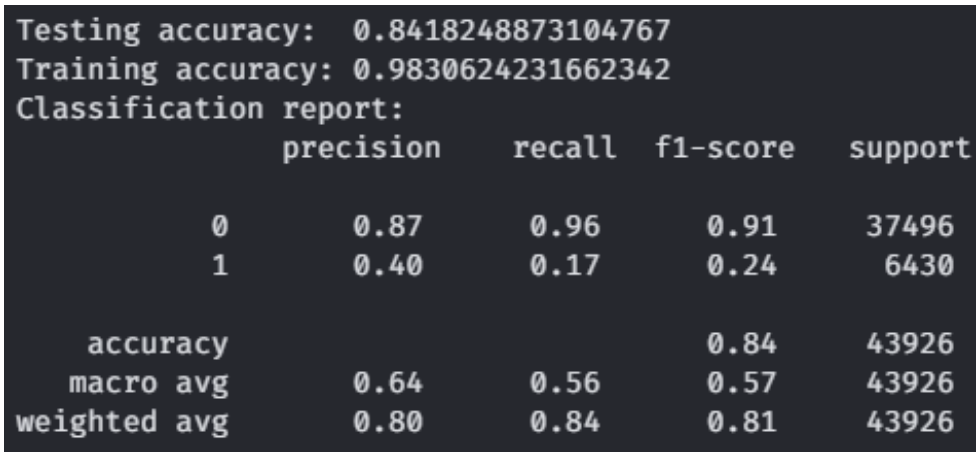

Figure 28. Random Forest Classification with feature from Spearman correlation and without hyperparameter tuning accuracy

Random Forest Classification with features from Spearman correlation analysis and with Hyperparameter Tuning accuracy:

```
Testing accuracy: 0.857328233847835
Training accuracy: 0.8618130492191413
Classification report:
              precision    recall  f1-score   support

           0       0.86      0.99      0.92     37496
           1       0.59      0.08      0.14      6430

    accuracy                           0.86     43926
   macro avg       0.73      0.54      0.53     43926
weighted avg       0.82      0.86      0.81     43926
```

Figure 29. Random Forest Classification with feature from Spearman correlation and with hyperparameter tuning accuracy

As observed in the preceding results, the difference in overall accuracy between models using Pearson and Spearman correlation analyses, with and without hyperparameter tuning, is very similar. The accuracy scores for both models without hyperparameter tuning are 0.84 and 0.85 with hyperparameter tuning. And the difference between the hyperparameter tuned and not hyperparameter tuned. The recall scores are quite low due to the class imbalance in the dataset. With more balanced data, the recall scores would likely improve significantly.

Table 2. Result Comparison

Metric/ Model	**Base**	**Pearson**		**Spearman**	
	Base	**Not Tuned**	**Tuned**	**Not Tuned**	**Tuned**
Weighted Precision	0.8040	0.8002	0.8216	0.8019	0.8232
Weighted Recall	0.8474	0.8410	0.8569	0.8418	0.8573
Weighted F1-Score	0.8121	0.8117	0.8070	0.8129	0.8080
Accuracy	0.8474	0.8409	0.8569	0.8418	0.8573

VII. Conclusion

From this study, it can be concluded that the use of a machine learning model to predict diabetes risk can be very helpful. Utilizing the ability of machine learning models with data mining methods will significantly help in early diagnosing diabetes risk. This study proves feature selection and hyperparameter tuning are crucial for improving the performance and accuracy of a classification model. The performance results of each Random Forest classifier model indicate that the influential factors in predicting diabetes risk are the features used and the hyperparameter tuning. Although the performance difference between the models with and without hyperparameter tuning is only 0.01, it is still a noteworthy distinction. Additionally, the Spearman and Pearson algorithms for correlation analysis produce very similar correlation results, leading to similar performance outcomes for both the Pearson and Spearman models, but both feature selection methods are recommended to reduce the number of dimensions of the data thus saving computing resources, simpler model complexity, and shorter run time.

VIII. Implications

Spearman and Pearson correlation analyses are essential for feature selection in building a model due to their ability to uncover relationships and dependencies within the dataset. By utilizing both methods, a comprehensive understanding of how features interact and influence each other can be obtained, leading to the selection of relevant and impactful features for the model. This approach helps in reducing dimensionality, improving model interpretability, and enhancing the model's predictive power by focusing on the most informative features while disregarding redundant or less influential ones.

The study emphasizes the significant impact of hyperparameter tuning on improving model performance, including accuracy, precision, and recall. Tuned models demonstrate better generalization to unseen data, reduce overfitting, optimize computational resources, and provide guidelines for future model development. This highlights the crucial role of hyperparameter tuning in developing reliable and efficient machine learning models, influencing both research practices and industry standards. In all model implementations, it is advisable to use feature selection through correlation analysis and perform hyperparameter tuning. These steps help optimize the model to predict outputs with better generalization, reduce overfitting, and enhance convergence speed.

References

- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10107388/
- https://www.sciencedirect.com/science/article/pii/S2665917422002392
- https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset