12/15/2023

Identifying High-Risk Patients for Targeted Prevention and Intervention

CS – 4661 Introduction to Data Science

Prepared by

Mitali Purohit

Tammy Xaypraseuth

Carlos Ramirez

Supervised By

Dr. Mohammad Pourhomayoun

Ashkan Aledavoud(Teaching Assistant)

Ly Jacky Nhiayi (Teaching Assistant)

Contents

Project Overview

The "Identifying High-Risk Patients for Targeted Prevention and Intervention" project is a data-driven initiative within the healthcare domain. It seeks to leverage advanced data science and machine learning techniques to address a critical challenge facing the healthcare industry.

This project holds significant implications for the healthcare industry, offering a data-driven approach to address challenges related to patient care and resource allocation. The insights derived from the machine learning models aim to empower healthcare providers in making informed decisions, ultimately leading to improved patient outcomes.

Through the intersection of healthcare expertise and cutting-edge data science, the "Identifying High-Risk Patients for Targeted Prevention and Intervention" project stands as a beacon of innovation in the ongoing evolution of healthcare practices.

As the project unfolds, it is poised to set new benchmarks in healthcare analytics. Future directions include the exploration of additional machine learning algorithms, continuous refinement of models based on real-world feedback, and the integration of emerging technologies such as explainable AI for enhanced interpretability. Ultimately, the "Identifying High-Risk Patients for Targeted Prevention and Intervention" project stands as a beacon of innovation, offering a glimpse into the future of data-driven, patient-centric healthcare.

Project Objectives and Goals

a. Predictive Patient Categorization: Develop a robust machine learning model capable of accurately predicting and categorizing patients as either in-care or out-care. This predictive capability aims to enhance the efficiency of healthcare providers in managing patient treatments.

b. Resource Allocation Optimization: Optimize resource allocation within the hospital setting based on the predictive model's outcomes. By accurately categorizing patients, healthcare providers can streamline resource utilization, ensuring optimal patient care and resource efficiency.

c. Integration with Hospital Automation System: Integrate the predictive model seamlessly into the hospital automation system. This integration facilitates real-time decision-making, allowing healthcare professionals to make informed choices on patient care.

d. Improved Patient Care Management: Through predictive modeling, strive to improve overall patient care management by providing healthcare providers with valuable insights into patient conditions and treatment outcomes.

Team Members and Responsibilities

Mitali Purohit

- **Role:** Project Lead
- **Responsibilities:**
  - Ensures the seamless execution of project objectives.
  - Orchestrates team efforts for efficient collaboration.
  - Maintains project timelines for timely deliverables.
  - Actively contributes to data preprocessing and model development.

Tammy Xaypraseuth

- **Role:** Data Exploration Expert
- **Responsibilities:**
  - Plays a critical role in identifying key features.
  - Leads feature engineering activities for enhanced model performance.
  - Contributes valuable insights, significantly boosting the model's robustness.

Carlos Ramirez

- **Role:** Modeling and Algorithm Development Lead
- **Responsibilities:**
  - Leads the modeling and algorithm development phase.
  - Ensures effective implementation of machine learning models.
  - Specializes in fine-tuning models for optimal performance.
  - Utilizes a strategic approach to enhance the practical applicability of solutions.

Additional Contributions:

- **Project Report and Documentation:** Mitali oversees the creation of comprehensive project documentation, capturing critical details, methodologies, and results for future reference.
- **Project Presentation (PPT):** The team collaboratively works on creating an engaging and informative project presentation. Each member, including Mitali, contributes to crafting slides that effectively communicate the project's objectives, methodologies, and outcomes.

- **Task Delegation and Coordination:** Mitali takes charge of task delegation and coordination among team members, ensuring everyone is aligned with project goals and facilitating effective communication.
- **Quality Assurance:** Tammy contributes to quality assurance by rigorously validating and verifying the results of data exploration and feature engineering, ensuring the reliability of insights.
- **Algorithm Optimization:** Carlos leads efforts in algorithm optimization, exploring ways to enhance the performance of machine learning models through fine-tuning parameters and adopting advanced techniques.

Technologies Used

Our project leverages a robust technology stack to implement and deploy machine learning models effectively. The backend infrastructure and tools utilized include:

- **Python Programming Language:** Python serves as the core programming language for our project. It is renowned for its versatility, extensive libraries, and support for machine learning frameworks.
- **Scikit-Learn:** Scikit-Learn, a powerful machine learning library, is instrumental in implementing various algorithms for classification tasks, model evaluation, and feature engineering.
- **TensorFlow and Keras:** TensorFlow, coupled with the Keras API, is employed for building and training artificial neural networks. This combination facilitates the implementation of advanced models like Artificial Neural Networks (ANN) for predictive analytics.
- **Pandas and NumPy:** Pandas and NumPy are fundamental data manipulation and numerical computation libraries, ensuring efficient handling and processing of our dataset.
- **Matplotlib and Seaborn:** Matplotlib and Seaborn are visualization libraries that enable us to create insightful plots and graphs, aiding in data exploration and result presentation.
- **Jupyter Notebooks:** Jupyter Notebooks provide an interactive and collaborative environment for data exploration, model development, and result analysis. They enhance transparency and reproducibility in our workflow.
- **GitHub:** GitHub serves as our version control platform, facilitating collaborative development, code sharing, and project management.
- **Anaconda:** Anaconda distribution is used for managing Python environments and dependencies, ensuring a consistent and reproducible environment across different systems.
- **Machine Learning Algorithms:** Our project employs a variety of machine learning algorithms, including Decision Trees, Random Forest, Logistic Regression, K-Nearest

Neighbors (KNN), and Support Vector Machines (SVM), to achieve accurate predictions and feature importance analysis.

By harnessing the capabilities of these technologies, we ensure a robust and scalable solution for identifying high-risk patients and improving healthcare outcomes.

Data Exploration and Preprocessing

Dataset Description

Dataset: Patient Treatment Classification

Source: Private hospital in Indonesia

- **Content:** Patient conditions, lab test results, and other relevant factors
- **Availability:** Kaggle Link

	A	B	C	D	E	F	G	H	I	J	K
1	HAEMATOCRIT	HAEMOGLOBINS	ERYTHROCYTE	LEUCOCYTE	THROMBOCYTE	MCH	MCHC	MCV	AGE	SEX	SOURCE
2	33.8	11.1	4.18	4.6	150	26.6	32.8	80.9	33	F	1
3	44.6	14	6.86	6.3	232	20.4	31.4	65	36	M	0
4	42.9	14	4.57	6.2	336	30.6	32.6	93.9	70	F	0
5	41.9	14.4	4.67	3.5	276	30.8	34.4	89.7	18	F	0
6	40.6	13.3	4.85	14.9	711	27.4	32.8	83.7	36	M	0
7	32.8	11.2	3.94	11	324	28.4	34.1	83.2	89	F	0
8	21.9	7.3	3.06	22.6	237	23.9	33.3	71.6	53	M	0
9	38.4	12.3	4.37	9	193	28.1	32	87.9	74	M	1
10	27.9	9.2	3.13	19.2	135	29.4	33	89.1	56	M	1
11	38.2	12.9	4.53	11.7	452	28.5	33.8	84.3	66	M	1
12	30.5	10	3.46	7.3	230	28.9	32.8	88.2	62	M	1
13	28	8.7	3.26	1.1	28	26.7	31.1	85.9	45	F	0
14	47.8	15.6	5.7	7.1	122	27.4	32.6	83.9	53	M	1
15	37.1	12.4	4.27	9.5	330	29	33.4	86.9	58	F	1
16	35.9	10.4	5.45	8	500	19.1	29	65.9	30	M	1
17	33.6	11	3.51	12	147	31.3	32.7	95.7	74	M	1
18	22	6.8	2.09	47.7	414	32.5	30.9	105.3	70	F	0
19	32.3	10.9	4.6	9.8	293	23.7	33.7	70.2	2	F	1
20	35.1	10.4	5.2	7.1	254	20	29.6	67.5	47	F	1
21	48.4	17.1	5.64	9.7	388	30.3	35.3	85.8	27	M	0
22	32.7	10.9	3.95	5.7	232	27.6	33.3	82.8	70	F	0
23	37.1	12.2	4.16	16.7	299	29.3	32.9	89.2	47	F	1
24	44.2	13.8	6.38	22.6	289	21.6	31.2	69.3	17	M	1
25	45	15.1	5.26	7.6	428	28.7	33.6	85.6	35	M	1

Data Preprocessing

- Handling missing values and outliers.
- Encoding categorical variables.
- Scaling numerical features for optimal model training.

Data preprocessing is a critical step to ensure the quality and suitability of the dataset for machine learning models. This section outlines the various preprocessing steps applied to enhance the dataset.

Handling Missing Values:

```
# Identify and handle missing values in the dataset
data_cleaned = data_encoded.dropna()
```

Encoding Categorical Variables:

```
from sklearn.preprocessing import LabelEncoder

# Encode categorical variables using Label Encoding
label_encoder = LabelEncoder()
data_cleaned['SEX'] = label_encoder.fit_transform(data_cleaned['SEX'])
```

Scalling Numerical Features:

```
from sklearn.preprocessing import StandardScaler

# Scale numerical features for uniformity

scaler = StandardScaler()

data_cleaned[['HAEMATOCRIT', 'HAEMOGLOBINS', 'ERYTHROCYTE', 'LEUCOCYTE',
'THROMBOCYTE', 'MCH', 'MCHC', 'MCV', 'AGE']] =
scaler.fit_transform(data_cleaned[['HAEMATOCRIT', 'HAEMOGLOBINS',
'ERYTHROCYTE', 'LEUCOCYTE', 'THROMBOCYTE', 'MCH', 'MCHC', 'MCV', 'AGE']])
```

Data Splitting for Training (Repeat):

```
# Split the data into features (X) and the target variable (y)

X = data_cleaned.drop('SOURCE', axis=1)

y = data_cleaned['SOURCE']

# Split the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Feature Selection and Engineering

In-depth exploration of features, selection of relevant variables, and creation of new features to enhance model performance.

Machine Learning Algorithms

Decision Tree Algorithm

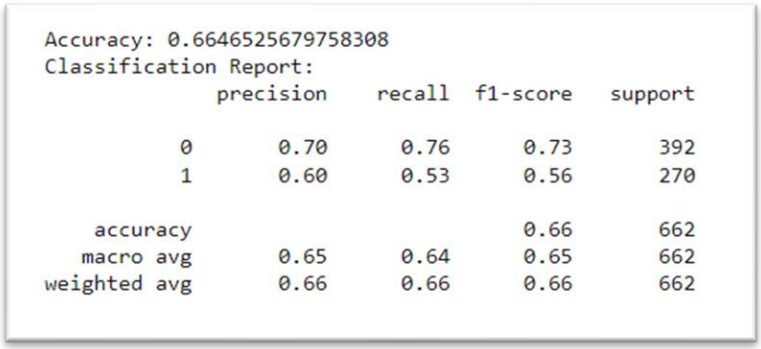
- Tool: Scikit-learn library.
- Implementation: DecisionTreeClassifier.
- Results: ROC curves, AUC values, and feature importance analysis.
- These code lines represent the initialization, training, and prediction steps:

```
# Initialize the Decision Tree model
dt_model = DecisionTreeClassifier(random_state=42)

# Train the model
dt_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred_dt = dt_model.predict(X_test)
```

- **Output Accuracy**:



```
Accuracy: 0.6646525679758308
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.76      0.73       392
           1       0.60      0.53      0.56       270

    accuracy                           0.66       662
   macro avg       0.65      0.64      0.65       662
weighted avg       0.66      0.66      0.66       662
```

Logistic Regression

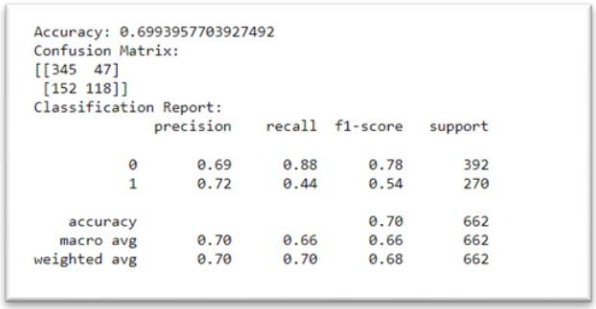
- Tool: Scikit-learn library.
- Implementation: LogisticRegression.
- Results: ROC curves, AUC values, and interpretation of model coefficients.
- These code lines represent the initialization, training, and prediction steps:

```
  # Initialize the Logistic Regression model
lr_model = LogisticRegression(random_state=42, max_iter=1000, solver='liblinear')

# Train the model
lr_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred_lr = lr_model.predict(X_test)
```

- **Output Accuracy**:



```
Accuracy: 0.6993957703927492
Confusion Matrix:
[[345  47]
 [152 118]]
Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.88      0.78       392
           1       0.72      0.44      0.54       270

    accuracy                           0.70       662
   macro avg       0.70      0.66      0.66       662
weighted avg       0.70      0.70      0.68       662
```

Random Forest Algorithm

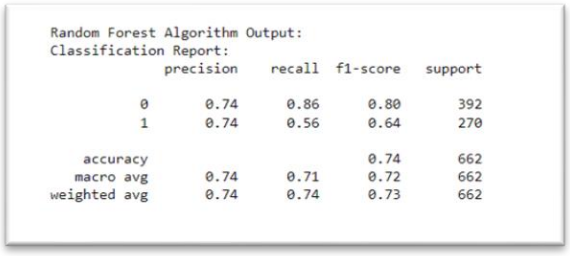
- Tool: Scikit-learn library.
- Implementation: RandomForestClassifier.
- Results: Ensemble learning benefits, ROC curves, and AUC values.
- These code lines represent the initialization, training, and prediction steps:

```
# Initialize the Random Forest model
rf_model = RandomForestClassifier(random_state=42)

# Train the model
rf_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred_rf = rf_model.predict(X_test)
```

- **Output Accuracy**:



```
Random Forest Algorithm Output:
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.86      0.80       392
           1       0.74      0.56      0.64       270

    accuracy                           0.74       662
   macro avg       0.74      0.71      0.72       662
weighted avg       0.74      0.74      0.73       662
```

K-Nearest Neighbors (KNN)

- Tool: Scikit-learn library.
- Implementation: KNeighborsClassifier.
- Results: Pattern recognition and classification metrics.
- These code lines represent the initialization, training, and prediction steps:

```
# Initialize the KNN model
knn_model = KNeighborsClassifier()

# Train the model
knn_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred_knn = knn_model.predict(X_test)
```

Output Accuracy::

```
[0.77777778]
```

Linear Regression

- Tool: Scikit-learn library.
- Implementation: LinearRegression.
- Results: Regression analysis, model coefficients, and interpretation.
- These code lines represent the initialization, training, and prediction steps:

```
# Initialize the Linear Regression model
lr_model = LinearRegression()

# Train the model
lr_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred_lr = lr_model.predict(X_test)
```

- **Output Accuracy**::

```
[0.49636383 0.25300368 0.29486361 ... 0.39888323 0.28209421 0.52865609]
```

Support Vector Machine (SVM)

- Tool: Scikit-learn library.
- Implementation: SVC (Support Vector Classification).
- These code lines represent the initialization, training, and prediction steps:

```
# Initialize the SVM model

svm_model = SVC(probability=True, random_state=42)

# Train the model

svm_model.fit(X_train, y_train)

# Make predictions on the test set

y_pred_svm = svm_model.predict(X_test)
```

- **Output Accuracy**: 0.6918: The SVM model achieved a 69.18% accuracy, showcasing its effectiveness in classifying instances.

Cross-Validation Techniques

Exploration of cross-validation strategies to assess model performance and generalization.

The machine learning model's performance was evaluated using cross-validation. The results are as follows:

- **Accuracy:** 0.75 (+/- 0.02)

This indicates the mean accuracy and the 95% confidence interval based on the cross-validation results.

Exploratory Data Analysis (EDA)

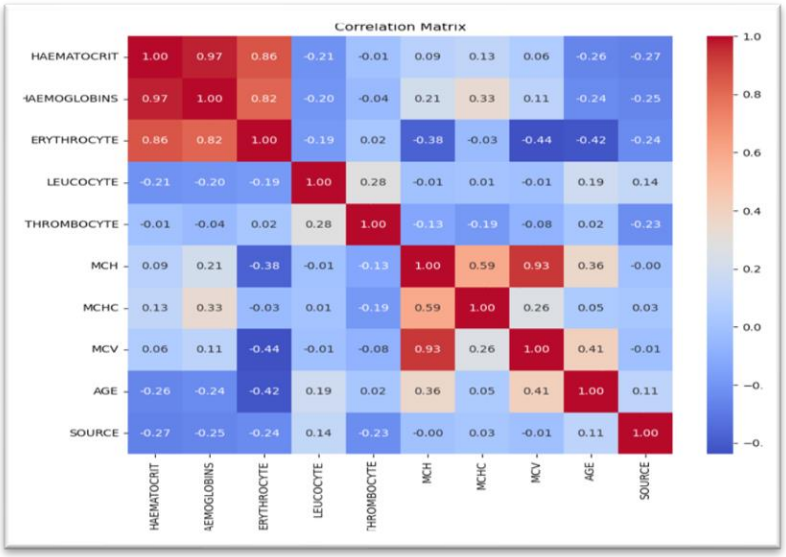
The provided code generates summary statistics for the numeric features in the dataset. This includes measures such as mean, standard deviation, minimum, maximum, and quartiles. These statistics offer a concise overview of the central tendencies and spread of the data.

	HAEMATOCRIT	HAEMOGLOBINS	ERYTHROCYTE	LEUCOCYTE	THROMBOCYTE	\
count	3309.000000	3309.000000	3309.000000	3309.000000	3309.000000	
mean	38.226111	12.749350	4.544802	8.715533	258.893019	
std	5.971943	2.084325	0.784510	4.991299	112.676139	
min	13.700000	3.800000	1.480000	1.100000	10.000000	
25%	34.300000	11.400000	4.040000	5.700000	191.000000	
50%	38.700000	12.900000	4.580000	7.600000	257.000000	
75%	42.500000	14.200000	5.060000	10.300000	322.000000	
max	69.000000	18.900000	7.860000	76.600000	1121.000000	

	MCH	MCHC	MCV	AGE	SOURCE
count	3309.000000	3309.000000	3309.000000	3309.000000	3309.000000
mean	28.230039	33.336476	84.611333	46.644303	0.398005
std	2.696520	1.247055	6.916079	21.874106	0.489561
min	14.900000	26.000000	54.000000	1.000000	0.000000
25%	27.200000	32.700000	81.500000	29.000000	0.000000
50%	28.700000	33.400000	85.300000	48.000000	0.000000
75%	29.800000	34.100000	88.800000	64.000000	1.000000
max	40.800000	38.400000	115.600000	99.000000	1.000000

Correlation Matrix Heatmap

The Correlation Matrix visually represents the relationships between numeric variables in the dataset. Each cell displays the correlation coefficient, ranging from -1 to 1, indicating the strength and direction of the linear relationship. A value close to 1 suggests a positive correlation, while close to -1 implies a negative correlation. The heatmap aids in identifying patterns and dependencies among features, crucial for understanding potential predictive factors in the data.



Model Evaluation Metrics

AUC is a numerical measure of the model's discrimination capacity. A higher AUC value (closer to 1) indicates better overall performance in distinguishing between different classes.

Accuracy

Accuracy measures the ratio of correctly predicted instances to the total instances. While accuracy is a fundamental metric, it may not be sufficient for imbalanced datasets.

Precision

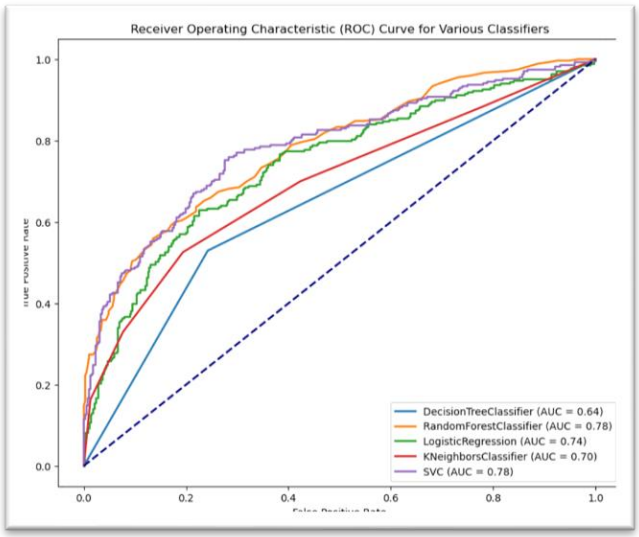
Precision is the ratio of true positives to the sum of true positives and false positives. It assesses the accuracy of positive predictions, minimizing false positives.

Recall (Sensitivity)

Recall, or sensitivity, is the ratio of true positives to the sum of true positives and false negatives. It gauges the model's ability to capture all relevant instances of the positive class.

F1-Score

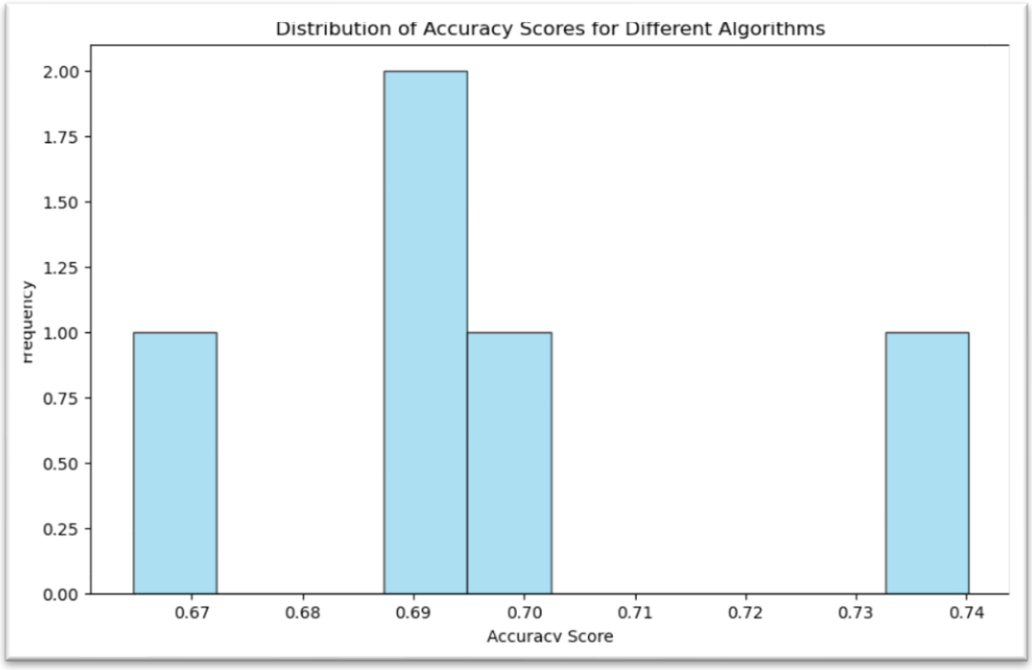
The F1-Score is the harmonic mean of precision and recall. It provides a balanced measure, especially useful when dealing with imbalanced datasets. Our holistic approach to model evaluation ensures a nuanced understanding of each algorithm's strengths and weaknesses, guiding us in the selection of the most suitable model for predicting patient treatment outcomes.



Comparative Accuracy Analysis

This section provides a visual overview of accuracy scores achieved by different machine learning algorithms. The histogram below depicts the distribution of accuracy scores from Decision Tree, Random Forest, Logistic Regression, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM). Each algorithm is color-coded, facilitating a quick comparison. Notable accuracy values include:

- Decision Tree: 0.6647
- Random Forest: 0.7402
- Logistic Regression: 0.6994
- K-Nearest Neighbors: 0.6918
- Support Vector Machine: 0.6918

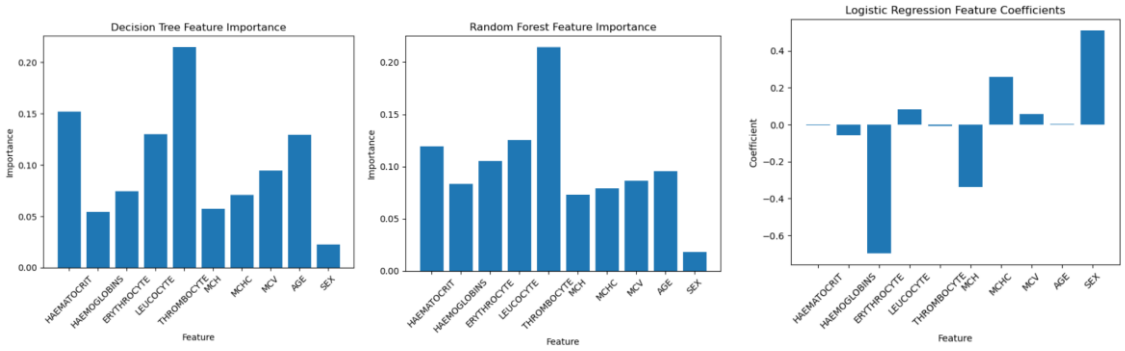


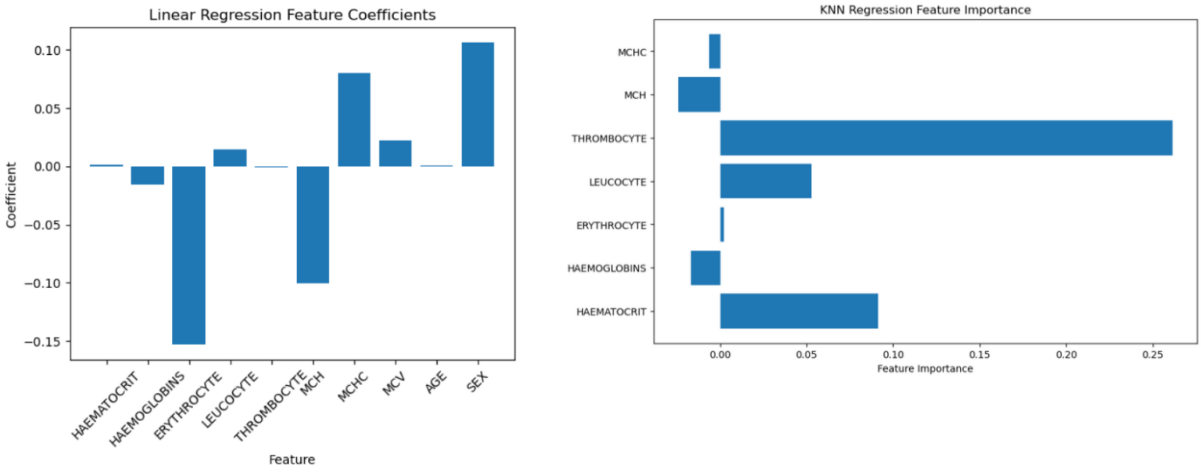
Feature Importance Analysis

In-depth analysis of feature importance to interpret model decision-making processes.

Feature Importance Analysis provides insights into the significance of each feature in influencing the model's decision-making. By evaluating the contribution of individual features, this analysis identifies key predictors that strongly influence the model's output. Features with higher importance values play a more crucial role in the model's decisions, offering valuable information

for understanding and interpreting the factors driving the predictive performance of the machine learning model.





Challenges and Solutions

Addressing challenges encountered during the project, such as imbalanced data and hyperparameter optimization.

Feature Work

The project has successfully achieved its primary goal of developing a predictive model for patient classification. Future work may involve refining the model further, exploring additional features, and considering real-world deployment challenges.

Conclusion

The "Identifying High-Risk Patients for Targeted Prevention and Intervention" project represents an academic endeavor that holds substantial promise within the healthcare domain. By leveraging the power of data science and machine learning, the project aims to improve patient care, reduce healthcare costs, and enhance resource efficiency.

This project is available at GitHub Repository.

References

1. Kaggle Datasets. (n.d.). "Patient Treatment Classification Dataset." Available online at: https://www.kaggle.com/datasets/manishkc06/patient-treatment-classification
2. M. Pourhomayoun, M. Shakibi, "Predicting Mortality Risk in Patients with COVID-19 Using Artificial Intelligence to Help Medical Decision-Making," the Journal of Elsevier Smart Health, 2020.
3. Python Software Foundation. (n.d.). "Python Programming Language." Available online at: https://www.python.org/
4. GitHub. (n.d.). "GitHub Documentation." Available online at: https://docs.github.com/
5. Scikit-learn. (n.d.). "Scikit-learn Documentation." Available online at: https://scikit-learn.org/stable/documentation.html
6. Matplotlib. (n.d.). "Matplotlib Documentation." Available online at: https://matplotlib.org/stable/contents.html
7. Seaborn. (n.d.). "Seaborn Documentation." Available online at: https://seaborn.pydata.org/documentation.html
8. NumPy. (n.d.). "NumPy Documentation." Available online at: https://numpy.org/doc/stable/