B.Tech. Project EndSem Report

Vreendar IT Solutions

ML Intern

Sheelampally Sai Shiva
202051173

Under the supervision of Pooja Kumari M (Off campus mentor)
Under the supervision of Dr.Kamal Kishore Jha (On campus mentor)

Declaration

I, Sheelampally Sai Shiva, declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referred to the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated any idea/data/fact/source in my submission. I fully understand that any violation of the above will invite a disciplinary action by the institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained.

Sheelampally Sai Shiva.202051173
Date:14-05-2024
Place: Banglore,Karnataka.

Figure 1: Pooja Kumari.M

Abstract

Before being offered tasks and assignments, all interns are required to complete raining programme to understand better about work culture. We are currently in the phase of training. We are given a topic to learn about each week and gradually tasks will be assigned.We get to meet with a mentor to talk about how we can get better.It containes the topics learned and how they can be implemented.

This endterm report encapsulates our project's scope, technology stack, progress in mastering concepts, and adherence to professional standards. Our project, focused on Anemia Analysis, utilized a technology stack comprising Python Libraries and machine learning concepts. Through a blend of theoretical learning and practical implementation aiming at enhancing skills and readiness for real-world challenges, underscored by our commitment to maintaining proficient guidelines and decorum. Moving forward, we remain dedicated to continuous improvement and embracing new challenges in the dynamic field.

Introduction

My internship is at Vreendhar IT Solutions, a company specializing in customer services, development, and management services, I had the opportunity to deeply engage with the field of machine learning. This hands-on experience allowed me to apply my theoretical knowledge in a practical setting and had a chance of collaborating with professionals and working with real-world problems. The startup environment cultivated a culture of innovation and teamwork, providing me with an ideal platform for learning and personal growth. I am enthusiastic about the opportunity to make significant contributions to projects that utilize cutting-edge technologies. I believe that this experience will play a pivotal role in shaping my career in machine learning.

Contents

Chapter 1

Week1:Induction and familiarity with work system and employees

Chapter 2

Week2:Python programming

One of the key reasons for Python's popularity is its vast array of libraries, which provide developers with access to a wide range of tools and functionality.

In the field of big data technology, Python is widely used, with libraries such as TensorFlow, Scikit-Learn, Pandas, NumPy, and Keras being particularly popular. These libraries provide powerful tools for data manipulation, analysis, and machine learning.

During our training and evaluation from SWE, we gained a solid understanding by revisiting Python's fundamentals, including its syntax and data structures, (OOP) concepts in Python and how to use Python collections effectively. Additionally, we gained practical experience with Python's robust data science libraries, which will be valuable in our future projects and endeavors. And as it was python we had tasks like code correction and review from SWE.

2.1 Introduction to Python and Python Basics

Python basics cover fundamental concepts such as variables, data types, operators, control flow (if statements, loops), functions, and basic input/output operations. Understanding these concepts is essential for writing Python programs and is the foundation for more advanced topics in Python programming.

2.2 Python Data Structures

Python offers a variety of data structures to efficiently store and manipulate data. These include lists, tuples, sets, and dictionaries, each with its own characteristics and use cases. Understanding these data structures is essential for effective Python programming and ML.

2.3 Object-Oriented Programming in Python

Object-oriented programming (OOP) is a programming paradigm that uses objects and classes to organize code. Python supports OOP principles, allowing developers to create reusable and modular code. Topics covered in this section include classes, objects, inheritance, polymorphism, and encapsulation.

Chapter 3

Week 3 :Data Manipulation and Data Visualisation

3.1 Pandas

Pandas is a Python library used for data manipulation and analysis. It provides data structures like DataFrame and Series, which are ideal for handling structured data such as CSV files or SQL tables

3.2 NumPy

NumPy is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

3.3 SciPy

SciPy is a library that builds on NumPy and provides additional functionality for scientific computing. It includes modules for optimization, integration, interpolation, and other tasks common in scientific and engineering applications.

3.4 Matplotlib

Matplotlib is a plotting library for Python that provides a MATLAB-like interface for creating static, animated, and interactive visualizations in Python. It is particularly useful for creating plots and charts to visualize data.

3.5 Seaborn

Seaborn is a data visualization library based on Matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics. It is particularly useful for visualizing complex datasets.

Chapter 4

Week 4-5:Machine Learning

Machine Learning a pivotal subset of artificial intelligence (AI), emphasizes creating algorithms and statistical models that empower computers to perform specific tasks by learning from data, without being explicitly programmed. This field hinges on the principle that machines can learn from information, discern patterns, and make informed decisions with little to no human intervention.

4.1 Data's Role

At the core of machine learning lies data, which can be categorized as structured, unstructured, or semi-structured. This data serves as the training ground for machine learning models. The algorithms' efficiency and accuracy are significantly influenced by both the quality and the volume of the data used.

4.2 Feature Importance

In the realm of machine learning, features represent the measurable attributes or characteristics within the data. The process of selecting significant features and aptly transforming them is vital for the development of potent machine learning models.

4.3 Algorithmic Foundation

Machine learning operates on algorithms—mathematical or computational methods—that analyze data to discover patterns and make informed predictions or decisions. These algorithms are broadly categorized into supervised learning, unsupervised learning, and reinforcement learning frameworks.

4.4 Supervised Learning Dynamics

This approach involves training models on a dataset that includes both inputs and corresponding target outputs. The aim is to map inputs to outputs, enabling the prediction of the target variable based on new input data.

4.5 Exploring Unsupervised Learning

Unsupervised learning trains models on data without predefined labels, aiming to uncover hidden structures or patterns within the data. Techniques such as clustering and dimensionality reduction are fundamental in this learning paradigm.

4.6 Reinforcement Learning Strategy

Reinforcement learning is characterized by an agent that learns decision-making through trial and error interactions with its environment. The focus is on maximizing cumulative rewards over time, which in turn shapes the agent's decision-making policy.

4.7 Model Evaluation Techniques

Evaluating the effectiveness and generalization capability of machine learning models is crucial. Commonly used metrics for this purpose include accuracy, precision, recall, F1 score, and the area under the receiver operating characteristic curve (AUC-ROC).

4.8 Addressing Overfitting and Underfitting

Overfitting happens when a model memorizes the training data, compromising its performance on new data. On the other hand, underfitting occurs when a model is too simple to capture the underlying structure of the data, resulting in poor performance on both training and unseen data.

4.9 Hyperparameter Optimization

Hyperparameter optimization is the process of fine-tuning an algorithm's hyperparameters, which dictate its learning process. This step is crucial for enhancing the model's performance by finding the optimal hyperparameter values.

4.10 Utilizing Cross-Validation

Cross-validation is a crucial method for assessing a model's performance. It involves splitting the data into multiple segments, training the model on some segments, and testing it on others. This technique helps predict how well the model will generalize to unseen data.

4.11 Grid search

Grid search is a method used in machine learning to systematically search through a predefined grid of hyperparameter values in order to optimize the performance of a model. By evaluating each combination of hyperparameters, usually through cross-validation, grid search helps identify the best set of parameters for the model. It is a key technique for hyperparameter optimization, essential for enhancing model accuracy and efficiency. Interns in machine learning programs often learn to implement grid search using libraries like scikit-learn, gaining valuable insights into hyperparameter tuning and model optimization.

Chapter 5

Week 6: Learning Flask

5.1 Introduction to Flask

Flask was introduced as a lightweight and user-friendly web framework for Python, suitable for beginners and small-scale projects.

5.2 Creating Web Applications with Flask

I learned the process of developing web applications with Flask, including routing, templates, forms, and other essential aspects.

5.3 Handling Requests and Responses

Understanding of how Flask handles HTTP requests and responses, including methods like GET, POST, and others.

5.4 Working with Databases

Integration of Flask with databases like SQLite, MySQL, or PostgreSQL for efficient data storage and retrieval.

5.5 Authentication and Authorization

Implementation of user authentication and authorization in Flask applications, covering sessions, cookies, and security best practices.

5.6 Deployment and Production

Deployment of Flask applications to production environments, focusing on scalability and performance considerations.

5.7 Building RESTful APIs

Creation of RESTful APIs using Flask, including endpoint design, request handling, and JSON response generation.

Chapter 6

Week 7: Learning FastAPI

6.1 Introduction to FastAPI

FastAPI was presented as a modern web framework for building APIs with Python 3.6+, known for its ease of use and rapid development capabilities.

6.2 Creating APIs with FastAPI

I learned how to use FastAPI to create APIs, including defining routes, request and response models, and handling authentication.

6.3 Data Validation and Serialization

Understanding of how FastAPI handles data validation and serialization, ensuring correct formatting of API requests and responses.

6.4 Asynchronous Programming

Exploration of FastAPI's support for asynchronous programming, enabling the development of efficient and responsive APIs.

6.5 Dependency Injection

Introduction to dependency injection in FastAPI and its role in managing dependencies and improving code modularity.

6.6 Testing FastAPI Applications

Best practices for testing FastAPI applications, including unit tests, integration tests, and testing of API endpoints.

6.7 Deployment and Production

Understanding of the deployment of FastAPI applications to production environments, with considerations for scalability and performance optimization.

6.8 Framework :Keras

In this week I have also learned about the Keras framework, a high-level neural networks API written in Python, designed for fast experimentation and prototyping of deep learning models. It provides a user-friendly interface with a modular architecture, allowing users to easily build and customize neural networks with minimal code. Keras seamlessly integrates with TensorFlow and other backend engines,.

6.9 Conclusion

The training program on development with Flask and FastAPI has equipped me with a strong foundation in creating web applications and APIs. I have learned how to integrate these skills into my machine learning work, allowing me to develop end-to-end solutions that encompass both web interfaces and machine learning models

Chapter 7

Week 8 -9: Machine Learning Model Research, Development, and Deployment

7.1 Introduction to development and depolyment

I have learned about the lifecycle of developing and deploying machine learning models, from research to production. This includes understanding best practices and tools to use along the way.

7.2 Understanding the Machine Learning Pipeline

Introduced about the typical steps involved in a machine learning pipeline, from data collection to model evaluation.

7.3 Working in the Jupyter Environment

Learned and got familarity with the Jupyter notebooks.

7.4 Transforming Jupyter Notebooks into Production Code

Discovered how to convert code from Jupyter notebooks into production-ready code, including best practices for writing clean and maintainable code.

7.5 Deploying Models and Serving Predictions

Learned how to deploy a machine learning model and serve predictions through an API, ensuring scalability and efficiency.

7.6 Creating a Python Package

Understood the process of packaging the code into a Python package for easy distribution and use.

7.7 Deploying into a Production Environment

Explored the considerations and steps involved in deploying a machine learning model into a realistic production environment.

7.8 Using Docker for Version Control

Learned how to use Docker to control software and model versions, ensuring Reproducibility and ease of deployment.

7.9 Adding a CI/CD Layer

Understood the importance of continuous integration and continuous delivery (CI/CD) in deploying machine learning models and how to implement it.

7.10 Ensuring Reproducibility

Learned the concept of reproducibility in machine learning model deployment and strategies to maximize it through versioning and code repositories.

7.11 Conclusion:Training

By the end of this training I have gained a comprehensive overview of the entire research, development, and deployment lifecycle of a machine learning model. I have understood the best coding practices and things to consider to put a model into production. I have also learned about the tools available to deploy my models and am well placed to take the deployment of the models in any direction that serves the needs of my organization.

Chapter 8

Form Week 11:Work Implementation:Introduction

Anemia is a medical condition characterized by a deficiency of healthy red blood cells in the body or a reduction in the amount of hemoglobin in the blood.Symptoms of anemia include fatigue, weakness, shortness of breath, dizziness, pale skin, irregular heartbeat, and headaches. Treatment for anemia depends on the underlying cause, but it may involve dietary changes, supplements, medication, or, in severe cases, blood transfusions.

It is important to identify and treat anemia promptly, as it can lead to complications such as heart problems, impaired cognitive function, and delayed growth and development in children.

8.1 Objective

Upon having a Discusiion with the team,team leaders and the customers Who had a specific goal for taking up an analysis of the blood anemic analysis we have decided To build up the machine leanring model to classifiy anemic condition and also to check how does gender division is having an impact on hemoglobin levels.And also to choose algorithm which would be work desired anylysis for detection in our case.

8.2 Exploratory Data Analysis

Data Overview:

The dataset comprises 1421 samples with six attributes: gender, hemoglobin, mean corpuscular hemoglobin (MCH), mean corpuscular hemoglobin concentration (MCHC), mean corpuscular volume (MCV), and result.
The result attribute is binary, representing non-anemic (0) and anemic (1) conditions. Gender is binary, and all other attributes are continuous variables.
The dataset seems clean with no missing values.
Class distribution: Non-Anemic (56.37), Anemic (43.63).

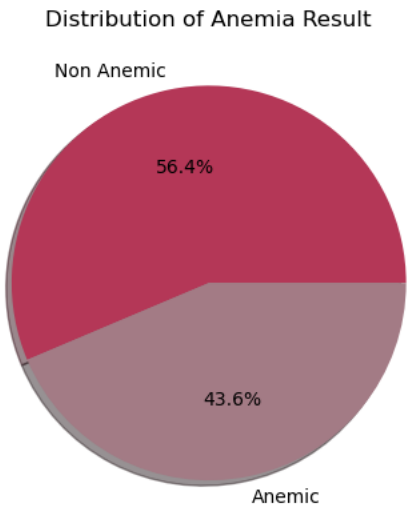


Figure 8.1: distirbution anemia

Class Imbalance: The longer the box, the more dispersed the data - Non anemic have more disperesed data.
Median line of a box plot lies outside of the box of a each other, there is likely to be a difference between the two groups.
For anemic we see data point that is located outside the whiskers of the box plot thus presence of outlier.
Addressing class imbalance is crucial as it can bias the predictive model towards the majority class.
Strategies like oversampling, undersampling, or using algorithms that handle class imbalance well should be considered.
Gender Impact on Anemia: Female participants have a higher anemia rate

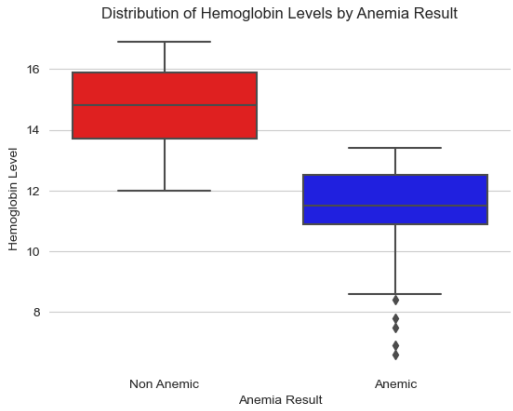


Figure 8.2: distirbution anemia

compared to males.
There are more female samples in the dataset (4.2 more than male).
Mean anemia rate: Male (31), Female (56).
Hemoglobin Levels:

Hemoglobin levels are essential for diagnosing anemia. The dataset shows a left-skewed distribution in hemoglobin levels.
Mean Hemoglobin Level: 13.41 g/dL
Highest: 16.90 g/dL, Lowest: 6.60 g/dL
Normal range: Men (13.2-16.6 g/dL), Women (11.6-15 g/dL).
MCH, MCHC, MCV:

Mean Corpuscular Hemoglobin (MCH): Average quantity of hemoglobin in a single red blood cell. Mean Corpuscular Hemoglobin Concentration (MCHC): Average weight of hemoglobin in packed red blood cells. Mean Corpuscular Volume (MCV): Average volume of a red blood cell. Ranges provided for each metric.
Next Steps:

Address class imbalance through suitable techniques.
Utilizing appropriate algorithms for classification, considering the dataset characteristics.
Feature engineering: Deriving new features or transforming existing ones to enhance predictive performance.
Model evaluation: Useing proper evaluation metrics to assess model performance, considering the imbalanced nature of classes.

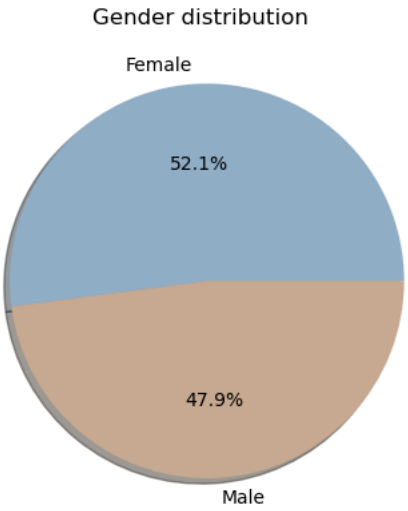


Figure 8.3: Gender distirbution anemia

By summarizing the EDA and outlining next steps, we can effectively proceed with building a machine learning model for anemia detection while considering the impact of gender and other relevant factors.
We need to explore/clean/filter it before processing it for machine learning It involves adding/deleting few columns or rows, joining some other data, and handling qualitative variables like dates.

8.3 Statistical Tests

We performed statistical tests including t-tests, odds ratio calculations, and chi-square tests for association between variables to identify significant relationships in the data.

8.3.1 t-test

We are using a t-test to determine whether there is a significant difference in the mean hemoglobin levels between males and females Here Hemoglobin have negaive skewness but t-test asumes have normal distribution. So before performing t-test, we would be taking the logarithm of the data, which can

help to reduce the skewness
T-Statistic: -0.41
P-Value: 0.679
Fail to reject null hypothesis: Gender has no impact on hemoglobin levels.
t-statistic is negative, it suggests that the mean hemoglobin level for males is slightly lower than the mean hemoglobin level for females, but the difference is not large.
P value is greater than the significance level of 0.05 so fail to reject null hypothesis.
Thus, we can conclude that there is not enough evidence to support the claim that gender has an impact on hemoglobin levels.

8.3.2 Odds Ratio

Odds ratio is a measure of the strength of association between two variables. We are looking for gender and Anemic condition(Result). The odds ratio tells us the odds of having anemia for females relative to males.

An odds ratio greater than 1 indicates that the odds of having anemia are higher for females, while an odds ratio less than 1 indicates that the odds of having anemia are higher for males. Optimization terminated successfully.
Current function value: 0.652524
Iterations 5
Odds Ratio for Gender: 2.86
An odds ratio of 2.86 means that females have 2.86 times the odds of being anemic compared to males. This indicates that being female is associated with a higher risk of anemia compared to being male, assuming that other factors in the model are held constant.

8.3.3 Chi-square test

Chi-square test is a statistical test used to determine whether there is a significant association between two categorical variables. In this case, we are using the chi-square test to determine whether there is a significant association between gender and anemia status.
Chi-Square Statistic: 90.06
P-Value: 0.000

Reject null hypothesis: Gender and anemia status are dependent.
The chi-square statistic is 90.06 and the p-value is less than 0.001.

Since the p-value is less than the significance level of 0.05, we reject the null hypothesis that gender and anemia status are independent, and conclude that there is evidence of a relationship between gender and anemia status.

8.4 Feature Selection

Feature selection techniques such as SelectKBest and Extra Tree Classifier were employed to identify the most relevant features for model training.

8.4.1 SelectKBest

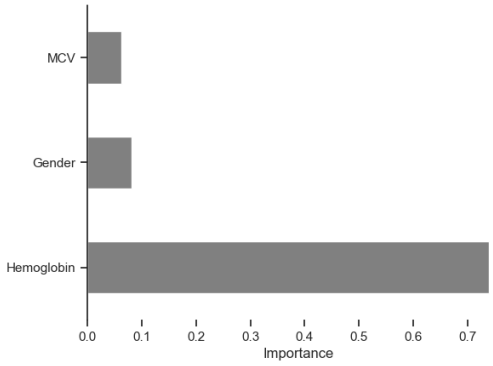
To validate the outcome , statistical method known as univariate selection using the SelectKBest method was used to select the top 3 features. SelectKBest uses Chi-squared as scoring functions i.e., tests whether the occurrences of a specific feature and a specific class are independent using their frequency distribution.

```
The best value of K is 2 with score 307.0244798773891.
---
        Specs       Score
0      Gender   43.648385
1  Hemoglobin  261.707512
2         MCH    0.803292
3        MCHC    0.212839
4         MCV    0.652452
---
        Specs       Score
1  Hemoglobin  261.707512
0      Gender   43.648385
2         MCH    0.803292
```

8.4.2 Extra Tree Classifier

When we analyze the relationship between one feature and the target variable, we ignore the other features. That is why it is called 'univariate Each feature has its test score. Finally, all the test scores are compared, and the features with top scores will be selected.

Cross-validation with feature selection relies on "Feature Importance" from tree-based models like Extra Trees Classifier, which assigns scores to each feature. Extra Trees' randomized feature selection makes it computationally efficient compared to Random Forest. Top features are extracted based on importance scores for predictive modeling.
Extremely Randomized Trees.



Gender and MCV are important feature. So we will be using these feature for our model.

8.5 Scaling features

Scaling features, such as through standardization or normalization, helps prevent features with larger scales from dominating the model's objective function. Standardization adjusts feature values to have a mean of zero and a standard deviation of one, ensuring consistent scales. Normalization scales feature values to a range between 0 and 1, useful when feature distributions are non-normal.

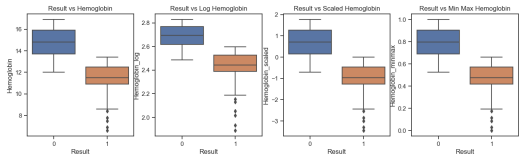
Taking the logarithm of a variable with skewed or extreme values compresses its range and down-weights extreme values, reducing the impact of outliers. Adding a small constant, like 0.01, before taking the logarithm prevents issues with zero or negative values, balancing distribution preservation and numerical stability.

8.5.1 Standardization

Standardization centers the values around the mean and ensures a unit standard deviation, resulting in a distribution with a mean of zero and consistent variance.

8.5.2 Scale Hemoglobin by Normalization

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.



On scaling we see the distrubution are quite similar to the original one. So we can say scaling is not as useful for this study.

```
# calculate Q1, Q3, and IQR of the 'anemia' column
Q1 = df['Hemoglobin_log'].quantile(0.25)
Q3 = df['Hemoglobin_log'].quantile(0.75)
IQR = Q3 - Q1

# print the values of Q1, Q3, and IQR
print('Q1:', Q1)
print('Q3:', Q3)
print('IQR:', IQR)
```

✓ 0.0s

```
Q1: 2.460443177609626
Q3: 2.7087166456453704
IQR: 0.24827346803574457
```

The log scaling effectively compresses the data distribution, particularly useful for left-skewed distributions. However, it assumes the data is positively skewed, where the tail extends more to the right. Standardization and normalization, while different in technique, both ensure consistent scaling and centering of the data, implying that the data may not require further scaling.

8.6 Splitting data into Training and Testing samples

We don't use the full data for creating the model. Some data is randomly selected and kept aside for checking how good the model is. This is known as Testing Data and the remaining data is called Training data on which the model is built. Typically 70percent of data is used as training data and the rest 30percent is used as testing data. Here I am keeping the selected feature only

8.7 Class Imbalance Handling

To address class imbalance, various techniques including random undersampling, random oversampling were applied to balance the distribution of classes in the training dataset.

8.7.1 Random Oversampling

Random oversampling duplicates examples from the minority class in the training dataset and can result in overfitting for some models.

```
# Oversampling with Logistic Regression
logreg = LogisticRegression()
logreg.fit(X_train_ros, y_train_ros)

y_pred_ros = logreg.predict(X_test)
[80] ✓ 0.0s

print('Accuracy :{0:0.5f}'.format(metrics.accuracy_score(y_test , y_pred_ros)))
print('AUC : {0:0.5f}'.format(metrics.roc_auc_score(y_test , y_pred_ros)))
print('Precision : {0:0.5f}'.format(metrics.precision_score(y_test , y_pred_ros)))
print('Recall : {0:0.5f}'.format(metrics.recall_score(y_test , y_pred_ros)))
print('F1 : {0:0.5f}'.format(metrics.f1_score(y_test , y_pred_ros)))
print('Kappa Statistic : {0:0.5f}'.format(cohen_kappa_score(y_test, y_pred_ros)))

[81] ✓ 0.0s
... Accuracy :0.99297
AUC : 0.99390
Precision : 0.98370
Recall : 1.00000
F1 : 0.99178
Kappa Statistic : 0.98565
```

8.7.2 Random undersampling

Random undersampling deletes examples from the majority class and can result in losing information invaluable to a model.

8.8 Data Leakage Handling

Precautions were taken to prevent data leakage by ensuring that information from the test set did not influence the training process. This involved careful separation of training and testing data and feature engineering strategies to avoid leakage.

8.9 Model Building:1

We check the performance of imbalance dataset first and later we implement some techniques to balance the dataset and again check the performance of balanced dataset. Finally, we will compare each regression models performance. in predciting LR we got the values as
Accuracy :0.99532
AUC : 0.99593
Precision : 0.98907
Recall : 1.00000
F1 : 0.99451
Recall (True Positive Rate): percentage of all Anemic cases captured.

Precision: Out of all items labeled as Anemic, what percentage of them is actually Anemic
Accuracy: How correct the model is (misleading for Anemic/imbalanced data)
F1 score: combination of recall and precision into one metric. F1 score is the weighted average of precision and recall, taking BOTH false positives and false negatives into account. Usually much more useful than accuracy, especially with uneven classes.
Model predicted 183 cases as Anemic and 244 as non anemic from the test dataset.
There are originally 181 Anemic cases and our model predicted only 183 fraud transaction.

So the accuracy of our model should be 183/181 but it is wrong

There are 181 cases recognised as True Postive, means they are orignally anemic cases and our model precited them as anemic cases only.

True Negative - 244 (truely saying negative - non anemic cases correctly identified as non anemic)

True Postive - 181 (truely saying positive - anemic cases correctly identified as anemic)
False Negative - 0 (falsely saying negative - anemic transaction incorrectly identified as non anemic)
False Positive - 0 (falsely saying positive - non anemic transaction incorrectly identified as anemic)
So, 100 is the real accuracy of our model, which the Recall Score (here we have same accuracy and recall). So we have the emphasis on Recall score and F1 score to measure the performance of our model, not the accuracy.

8.10 Performance Evaluation

Performance was measured using metrics such as accuracy, area under the curve (AUC), precision, recall, F1 score, and kappa statistic. These metrics provided a comprehensive assessment of the model's predictive capabilities.

8.11 Model Building:2

Random Forest (RF)

```
# Random Forest (RF)
RFmodels = []

RFmodels.append(('RF imbalance', RandomForestClassifier(),X_train,y_train,X_test,y_test))
RFmodels.append(('RF Undersampling', RandomForestClassifier(),X_train_rus,y_train_rus,X_test,y_test))
RFmodels.append(('RF Oversampling', RandomForestClassifier(),X_train_ros,y_train_ros,X_test,y_test))

# Call function to create model and measure its performance
build_measure_model(RFmodels)
```

Gaussian Naive Bayes (NB)

Performance measures of various classifiers:

Gaussian Naive Bayes (NB)

```
# Gaussian Naive Bayes (NB)
NBmodels = []

NBmodels.append(('NB imbalance', GaussianNB(),X_train,y_train,X_test,y_test))
NBmodels.append(('NB Undersampling', GaussianNB(),X_train_rus,y_train_rus,X_test,y_test))
NBmodels.append(('NB Oversampling', GaussianNB(),X_train_ros,y_train_ros,X_test,y_test))

# Call function to create model and measure its performance
build_measure_model(NBmodels)
```

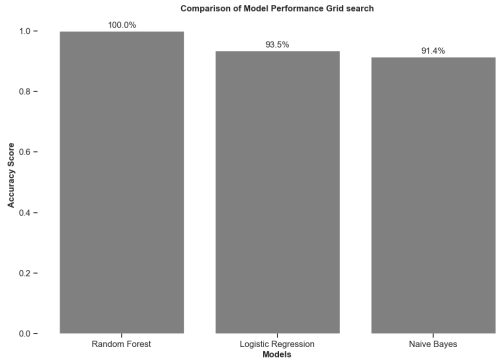
	Model	Accuracy_Test	AUC_Test	PrecisionScore_Test	RecallScore_Test	F1Score_Test	Kappa Stat
0	RF imbalance	1.000	1.000	1.000	1.000	1.000	1.000
1	RF Undersampling	1.000	1.000	1.000	1.000	1.000	1.000
2	RF Oversampling	1.000	1.000	1.000	1.000	1.000	1.000
3	RF SMOTE	1.000	1.000	1.000	1.000	1.000	1.000
4	KNN imbalance	0.979	0.980	0.962	0.989	0.975	0.957
8	KNN ADASYN	0.974	0.978	0.943	1.000	0.971	0.948
6	KNN Oversampling	0.974	0.976	0.952	0.989	0.970	0.948
7	KNN SMOTE	0.972	0.974	0.947	0.989	0.968	0.943
5	KNN Undersampling	0.970	0.972	0.942	0.989	0.965	0.938
9	NB imbalance	0.963	0.964	0.941	0.972	0.957	0.924
12	NB SMOTE	0.956	0.958	0.922	0.978	0.949	0.910
10	NB Undersampling	0.953	0.956	0.917	0.978	0.947	0.905
11	NB Oversampling	0.953	0.956	0.917	0.978	0.947	0.905

8.12 Hyperparameter Tuning

Hyperparameters of the selected algorithms were fine-tuned using GridSearchCV to optimize model performance.
It takes in a model, a grid of hyperparameters, and a scoring metric, and performs a cross-validated search over the hyperparameter grid to find the best combination. The result is the best combination of hyperparameters that produces the highest score on the chose.

```
RandomForestClassifier best params: {'max_depth': 3, 'n_estimators': 50} best score: 1.0
F1 score: 1.0
GaussianNB best params: {} best score: 0.9306075833714026
F1 score: 0.9565217391304347
```

8.13 Plotting Accuracy



8.14 Conclusion: Findings

Scaling of data with or without have no effect on the model performance. Based on hyperparameter tuning with GridSearchCV

RandomForestClassifier is best with 'max-depth': 3'n-estimators': 50 LogisticRegression with 'C': 10, 'solver': 'newton-cg'

RF performed well with and even withour handling class imbalance

Naive Bayes without balancing performed well compared with all class imbalance methods, with accuracy 96 and AUC 0.964, Kappa stat 0.924 and F1 score 0.957 For Anemia classification the important feature is to look for Hemoglobin, Gender and MCV.
Gender has no impact on hemoglobin levels with t-statistic is -0.41 and the p-value is 0.679.
Female is associated with a higher risk of anemia compared to male, assuming that other factors in the model are held constant with Odds Ratio for gender: 2.86.
Random forest in any balancing technique have both good capture True Positives and also False Positives. This means we capture more anemic cases So Random Forest with any balnacing technique is our final model, as this gives highest F1 score of 100 on test datasets.

Lastly the Random Forest model is to exported as pickle file:random-forest-model.pkl Which is to be used in Flask application. This internship gave me practical demonstration of ML concepts in Medical field

References

KAGGLE. https://www.kaggle.com/
GeeksforGeeks. (n.d.). *Machine Learning - GeeksforGeeks*. Retrieved from https://www.geeksforgeeks.org/machine-learning/
Tutorialspoint. (n.d.). *Machine Learning - Tutorialspoint*. Retrieved from https://www.tutorialspoint.com/machine_learning/index.htm

[2] Grammarly. (n.d.). *Grammarly - Grammar Check*. Retrieved from https://www.grammarly.com/grammar-check

Overleaf. (n.d.). *RGU Thesis Template 2024*. Retrieved from https://www.overleaf.com/latex/templates/rgu-thesis-template-2024/jjzgwddwzmqm