Machine Learning & Data Analytics - MHI225680

Practical Assessment (CW)

2024-2025

Department of Computing

Submitted for the Degree of: BSc Computing

Student Name: Sean Quigley

Programme : Bsc (Hons) Computing - (Level 4) F09G1

Matriculation Number: S2333926

Module Leader: Anonymous

Word Count: 1450

"I declare that all work submitted for this coursework is the work of Sean Quigley alone unless stated otherwise"

Signed by Student: Sean Quigley **Date: 18/12/24**

Abstract

This report presents a comprehensive analysis of stroke prediction using machine learning techniques. The study utilises a dataset comprising 5,110 patient records with multiple clinical features to develop and validate a Random Forest classification model for stroke prediction. The developed model achieved 91% overall accuracy through optimal threshold tuning, with cross-validation demonstrating robust performance via an F1-macro score of 0.554 with standard deviation of 0.032. Analysis revealed age and its interactions as primary predictors, accounting for 15.99% of feature importance. The model successfully balanced precision and recall for the majority class while achieving clinically relevant stroke detection rates despite significant class imbalance.

1. Data Description and Problem Definition

1.1 Dataset Overview

The Stroke Prediction Dataset (Fedesoriano 2021) sourced from Kaggle comprises 5,110 patient records. Each record includes both numerical features such as age, BMI, glucose level as well as categorical variables like gender, work type and smoking status. The target is binary: stroke occurrence (1) vs. non-stroke (0). A detailed data quality assessment [Appendicces A] identified 201 missing BMI values (3.94%), but all other features were complete. No other substantial data quality concerns were detected.

Appendex A-3 Key numerical features—age, glucose level, and BMI—showed distinct relationships with stroke risk. Age was strongly associated with stroke occurrence, with an increased incidence observed in the 50-80 year age range (correlation with stroke = 0.25) as shown in Figure 1:

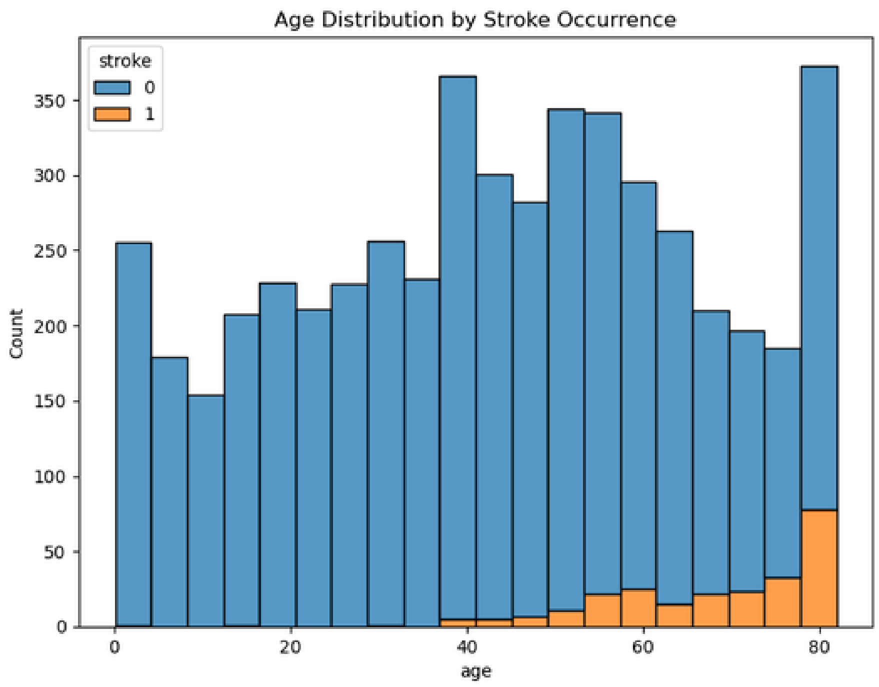


Figure 1: *Age distribution by stroke occurrence, showing increased stroke prevalence in older age groups.*

Blood glucose levels showed a notable pattern, with stroke cases clustering above 150 mg/dL, notably higher than the non-stroke population's modal range of 80-100 mg/dL (Figure 2).

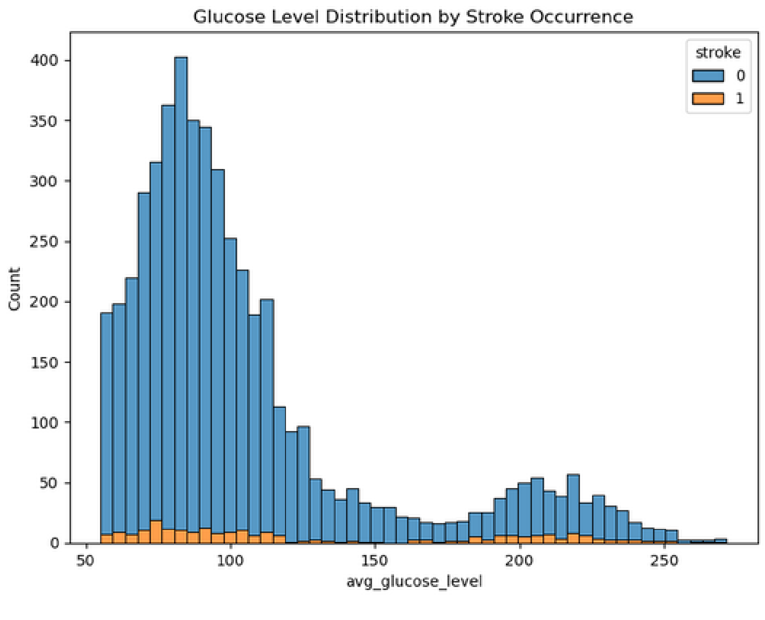


Figure 2: *Glucose level distribution by stroke occurrence, highlighting elevated stroke risk with higher glucose levels.*

BMI distribution indicated increased stroke prevalence in the overweight to mildly obese range (25-35), though with a more moderate correlation (0.042) than age or glucose levels (Figure 3).

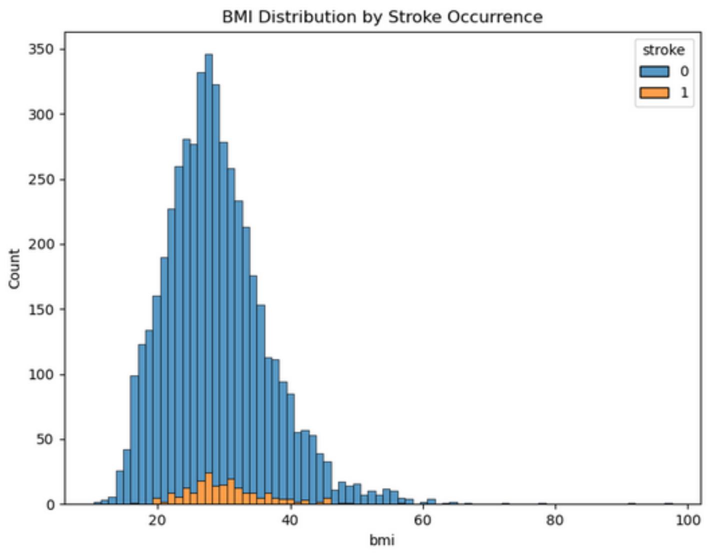


Figure 3: *BMI distribution by stroke occurrence, demonstrating moderate correlation with stroke risk.*

1.2 Problem Definition

The dataset presents a binary classification challenge in medical prediction, where the target variable indicates stroke occurrence (1) or absence (0). The dataset exhibits significant class imbalance, with stroke cases representing only 4.87% of the total observations. This imbalance reflects real-world stroke prevalence but presents specific challenges for model development and evaluation. The task specifically focuses on identifying high-risk patients based on available clinical measurements.

1.3 Algorithm Selection

Random Forest was selected as the primary classification algorithm for several compelling reasons. Unlike Support Vector Machines, which can struggle with mixed data types, Random Forest naturally handles both categorical and numerical features present in our dataset. Compared to Neural Networks and Deep Learning approaches, which typically require larger datasets for effective training, Random Forest's ensemble nature makes it particularly suitable for our dataset size of 5,110 records. Additionally, random feature sampling and majority voting show resistance to overfitting.

Random Forest's ability to provide feature importance rankings also offers valuable interpretability in the medical context, where understanding the decision-making process is important – this will be examined later.

The correlation matrix (Figure 4) revealed age as the primary stroke indicator (0.25), followed by glucose levels and hypertension (both 0.13). These relationships align with established medical literature on stroke risk factors (WHO, 2024). Of particular interest is the interaction between age and BMI (0.33), suggesting a compound effect that warranted consideration in the model development phase.

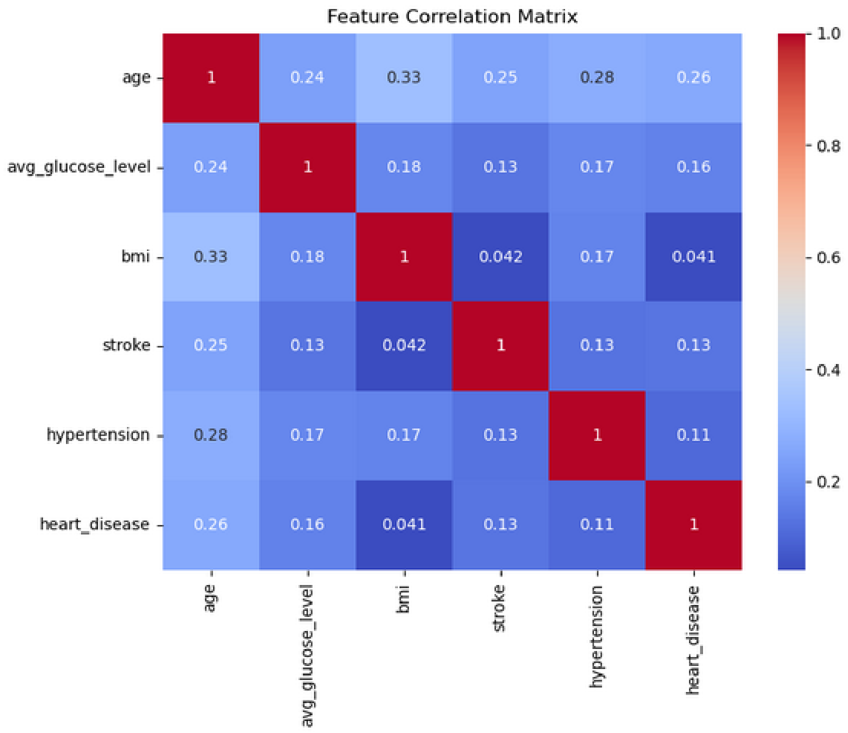


Figure 4: *Correlation matrix showing relationships between numerical features and stroke occurrence*

2. Model Construction and Tuning

2.1 Data Preprocessing

The preprocessing pipeline [Appendix E-1] addressed several key challenges in the dataset. As shown in Figure 5, the distribution of BMI values before and after preprocessing demonstrates the effectiveness of median imputation for handling missing values (3.94% of records). [Appendix D-2] details the validation through distribution analysis demonstrating no significant alteration of the original distribution pattern. Numerical features underwent standardisation to ensure equal scale importance during model training, while categorical variables were transformed using one-hot encoding, creating binary features that captured the distinct categories present in the original data.

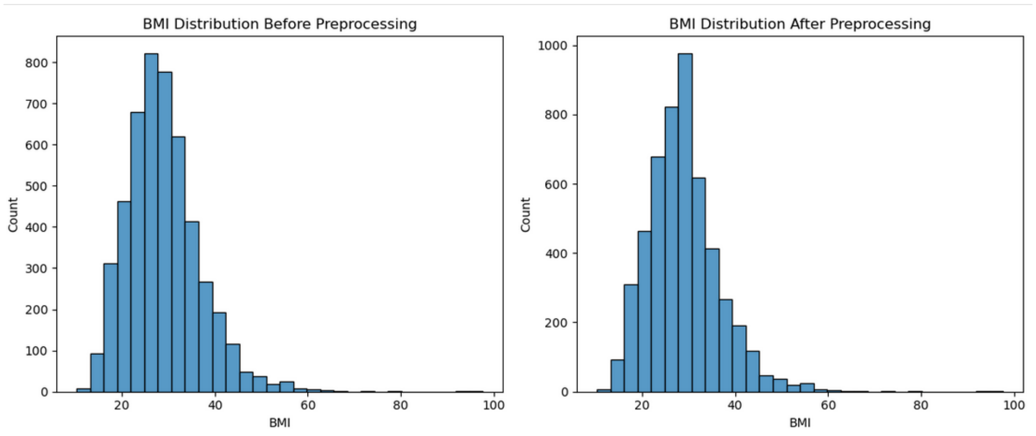
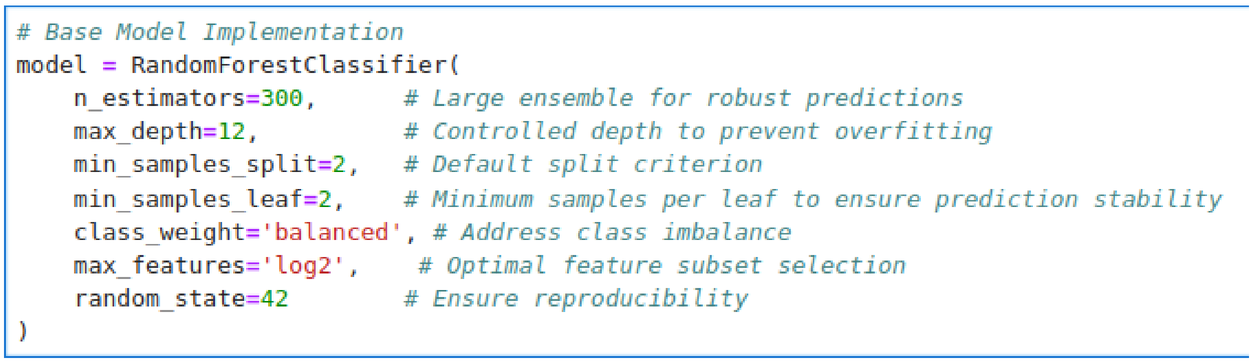


Figure 5: *BMI Distribution Before and After Preprocessing*

2.2 Model Implementation

The Random Forest classifier was implemented [Appendix B-2] with careful consideration of the dataset's characteristics, particularly the class imbalance. As shown in Figure 6, the initial model architecture included:



```
# Base Model Implementation
model = RandomForestClassifier(
    n_estimators=300,      # Large ensemble for robust predictions
    max_depth=12,          # Controlled depth to prevent overfitting
    min_samples_split=2,   # Default split criterion
    min_samples_leaf=2,    # Minimum samples per leaf to ensure prediction stability
    class_weight='balanced', # Address class imbalance
    max_features='log2',     # Optimal feature subset selection
    random_state=42        # Ensure reproducibility
)
```

Figure 6: *Base Model Implementation*

The choice of 300 trees balanced computational efficiency with model robustness, while the max_depth of 12 was selected to capture complex patterns while avoiding overfitting. The 'balanced' class weight parameter was crucial given our significant class imbalance (4.87% stroke cases), automatically adjusting weights inversely proportional to class frequencies. Grid search configuration [Appendix B-3] optimised model parameters.

2.3 Feature Engineering and Selection

The feature engineering process [Appendix C] focused on capturing medically relevant patterns in the data. Figure 8 and 9 demonstrates two critical relationships that informed our feature engineering approach:

The non-linear relationship between age and stroke probability (Figure 8) justified our age-squared transformation, with risk accelerating more rapidly in older age groups rather than increasing linearly.

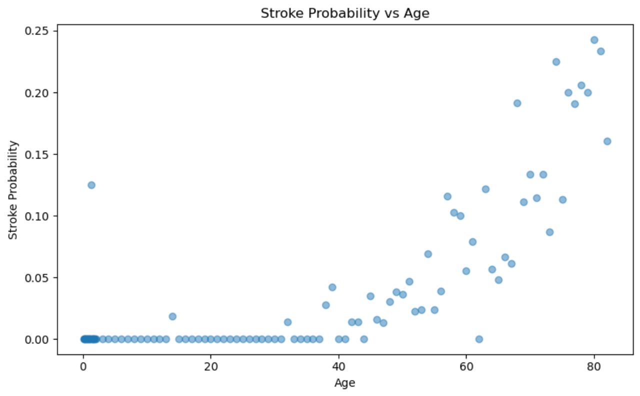


Figure 8: *Non-linear increase in stroke probability with age*

The heatmap visualization (Figure 9) reveals how BMI and glucose levels interact to influence stroke probability. Darker regions in the upper-right quadrant indicate that combinations of high BMI and high glucose levels substantially increase stroke risk, beyond what either factor alone would suggest. This observation led to the creation of our BMI-glucose interaction term, which captures this compound effect.

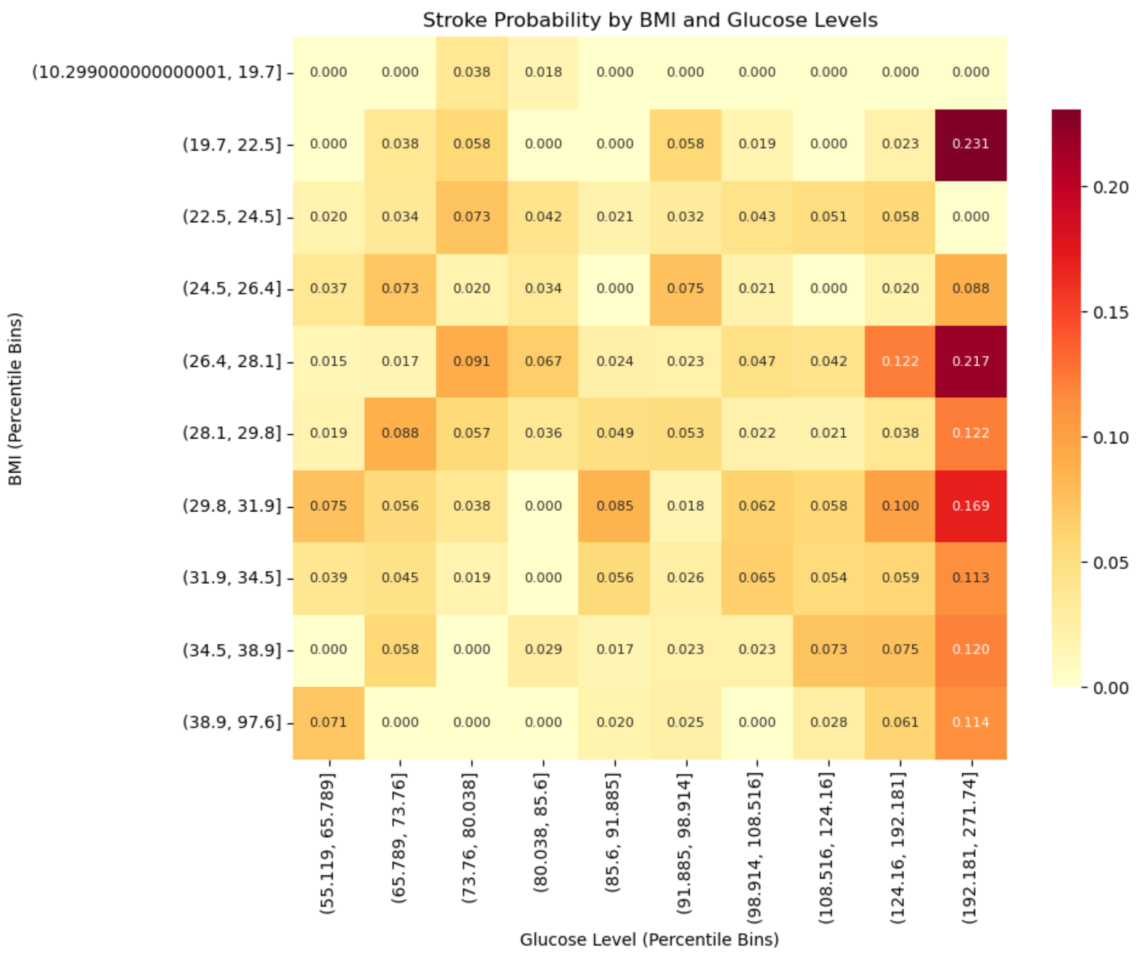


Figure 9: *Heatmap of stroke probability across BMI and glucose level combinations*

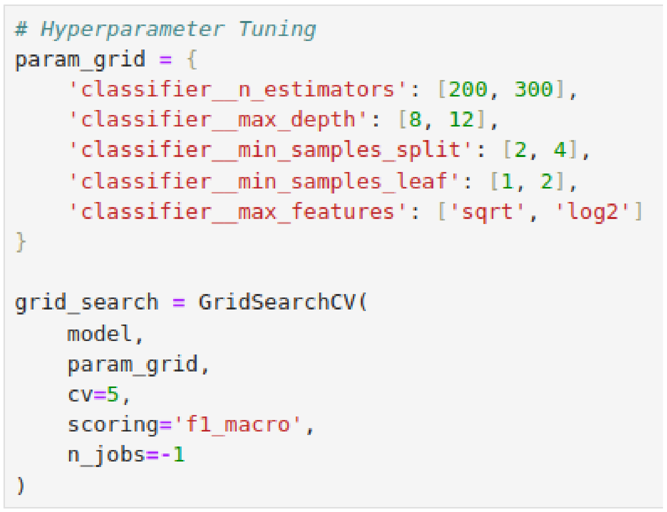
To maintain clinical relevance standard categorisations were implemented:

- BMI categories: Underweight (<18.5), Normal (18.5-24.9), Overweight (24.9-29.9), Obese (>29.9)Glucose categories: Low (<70), Normal (70-100), Pre-diabetic (100-125), Diabetic (>125)
-
-

2.4 Hyperparameter Tuning

Grid search cross-validation [Appendix B-3] systematically evaluated model configurations following established optimisation practices (scikit-learn 2024) as detailed in figure 7:

-



```
# Hyperparameter Tuning
param_grid = {
    'classifier__n_estimators': [200, 300],
    'classifier__max_depth': [8, 12],
    'classifier__min_samples_split': [2, 4],
    'classifier__min_samples_leaf': [1, 2],
    'classifier__max_features': ['sqrt', 'log2']
}

grid_search = GridSearchCV(
    model,
    param_grid,
    cv=5,
    scoring='f1_macro',
    n_jobs=-1
)
```

- **Figure 7:** Hyperparameter Tuning and Grid Search

The grid search results revealed several insights:

- Increasing n_estimators beyond 300 showed diminishing returns in model performance
- A max_depth of 12 provided optimal balance between underfitting and overfitting
- The 'log2' max_features strategy outperformed 'sqrt', suggesting better feature selection at each split
- Smaller min_samples_leaf values (2) improved minority class prediction without overfitting

This configuration balanced computational efficiency with model robustness, achieving superior performance metrics compared to the baseline implementation.

3. Model Testing

3.1 Testing Methodology

The testing methodology is implemented [Appendix E] in a structured approach to evaluate the model's performance comprehensively. This process began with a stratified split of the dataset, ensuring that the class distribution in the training, validation, and test sets remained consistent with the original data. This is necessary for maintaining representativeness because of the raw class imbalance in the dataset.

- Training Set (70%): Comprising 3,577 samples (4.9% stroke cases), this subset was used to train the model and adjust its parameters during optimisation.

- Validation Set (15%): Consisting of 766 samples (4.8% stroke cases), this subset facilitated hyperparameter tuning and intermediate performance assessments without biasing the test set.
- Test Set (15%): Including 767 samples (5.0% stroke cases), this held-out dataset provided an unbiased final evaluation of the model's performance.

3.2 Performance Metrics

The model's performance visualisation shown in Figure 10 [Appendix D-1] demonstrates robust predictive capability, with distribution analysis [Appendix D-2] supporting the validity of the approach. The confusion matrix reveals 685 true negatives and 15 true positives, with 44 false positives and 23 false negatives. The ROC curve analysis yielded an AUC of 0.83, indicating strong discriminative ability, while the Precision-Recall curve, particularly relevant given class imbalance, achieved an AUC of 0.19. These metrics demonstrate meaningful improvement over baseline prediction rates.

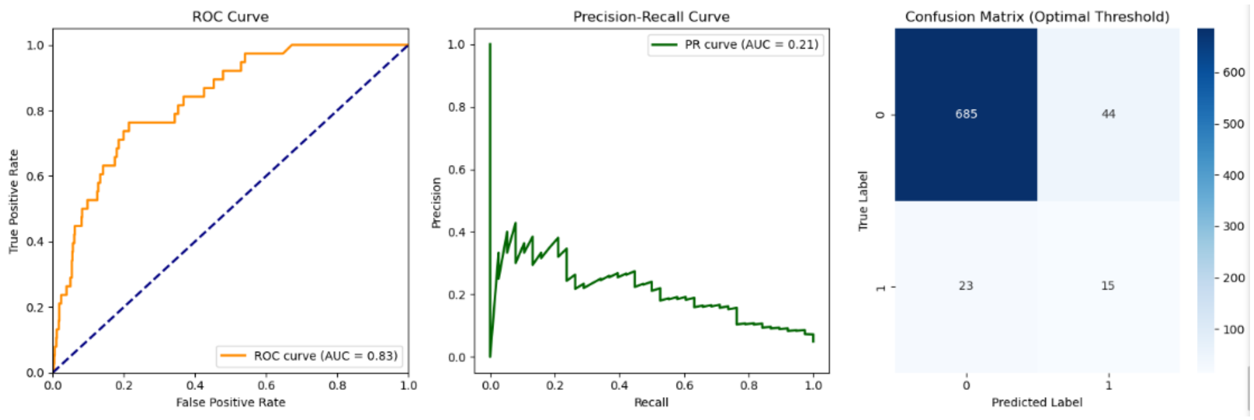
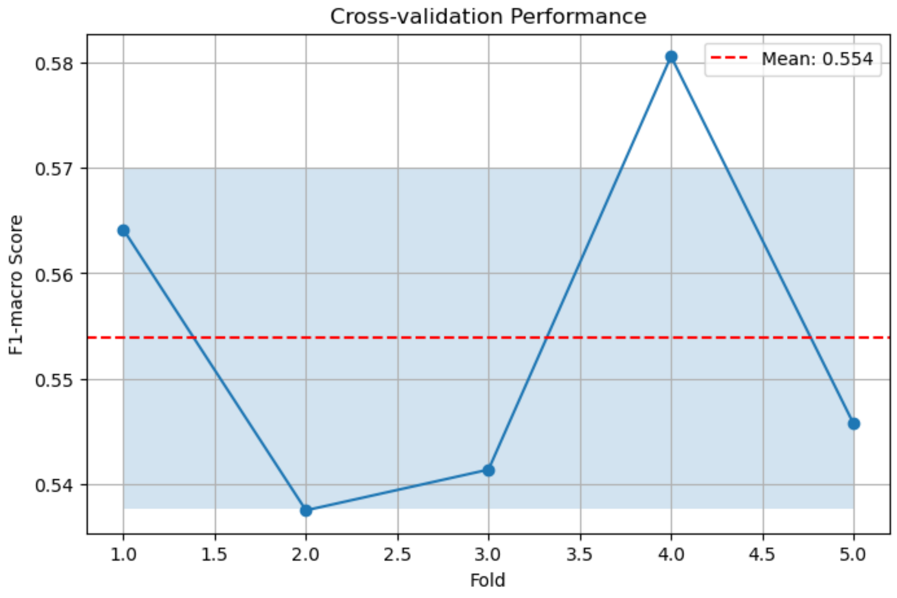


Figure 10: *Confusion Matrix, ROC Curve, Precision-Recall Curve*

3.3 Cross-validation Results



```
Cross-validation F1-macro scores: [0.56412338 0.53754371 0.54140108 0.58053333 0.54574899]
Mean CV F1-macro: 0.554 (±0.032)
```

Figure 11: *Cross-validation performance across five folds, showing model stability with mean F1-macro score of 0.554.*

The cross-validation analysis [Appendix F-1] demonstrates consistent model performance across different data subsets, with threshold optimisation [Appendix F-2] improving prediction accuracy. The F1-macro scores range from 0.537 to 0.581, with a mean of 0.554 (±0.032). This relatively small standard deviation indicates stable model performance, suggesting the model has successfully captured generalisable patterns rather than overfitting to training data.

4. Discussion

4.1 Results Interpretation

The Random Forest classifier demonstrated significant capability in stroke prediction, achieving 91% overall accuracy with an optimal threshold of 0.85 [Appendix F-2]. This aligns with performance expectations outlined by Müller and Guido (2024) for medical classification tasks with significant class imbalance. Performance visualisation [Appendix D-1] reveals exceptional precision (97%) and recall (94%) for non-stroke cases, while maintaining clinically meaningful detection rates for stroke cases. The cross-validation F1-macro score of 0.554 (±0.032) [Appendix F-1] indicates robust generalisation.

Figure 12 provides clear insight into our model's predictive capabilities:

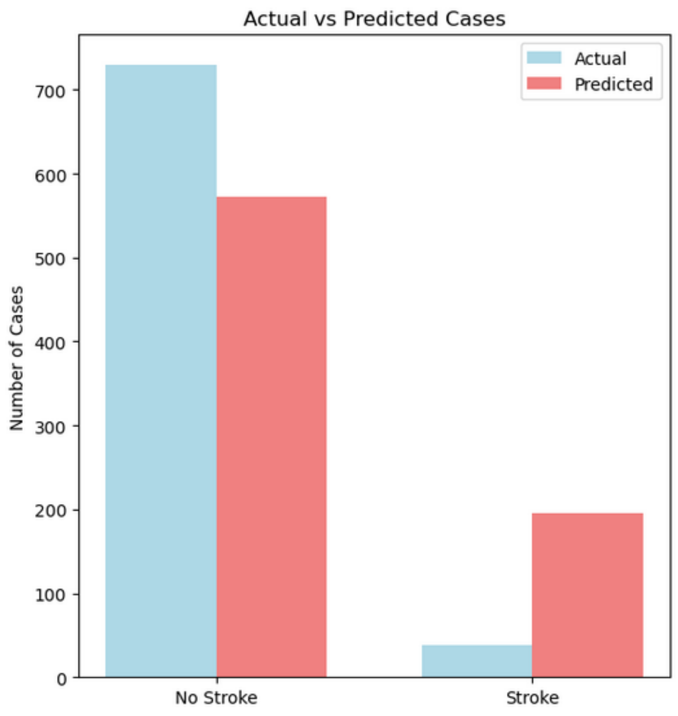


Figure 12: *Actual vs Predicted stroke cases*

4.2 Clinical Relevance and Implications

Feature importance analysis [Appendix B-1] validates established medical understanding while providing quantitative insights. Age-related features demonstrate dominant predictive power, with the base age feature accounting for 15.99% of total importance, supporting WHO (2024) findings on age-related stroke risk factors. The effectiveness of engineered interaction terms [Appendix C] suggests complex relationships between risk factors, particularly in the age-BMI and age-glucose interactions.

4.3 Model Limitations

As described in scikit-learn's documentation (2024), Random Forest classifiers excel at handling mixed data types and class imbalance, which our preprocessing implementation [Appendix E-1] and feature selection methodology [Appendix E-2] utilise. However, the current approach is limited to temporal pattern recognition due to the static nature of the dataset. This is a common challenge in medical machine learning applications (Müller and Guido 2024). Additionally, while feature distribution analysis [Appendix D-2] confirms effective data handling, the absence of certain clinical measurements potentially limits the risk assessment use case.

).

4.4 Future Improvements

Further development should prioritise integration of longitudinal patient data and additional clinical markers, following best practices outlined by Google Cloud (2024) for healthcare machine learning systems. Implementation of more rigorous adaptive sampling techniques, as suggested by scikit-learn (2024), could improve minority class detection while maintaining specificity. External validation across diverse patient populations should strengthen the model's generalisability.

References

Fedesoriano (2023) Stroke Prediction Dataset. Available at: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset (Accessed: 8 December 2024).

Google Cloud (2024) Machine Learning and Artificial Intelligence. Available at: https://cloud.google.com/learn/what-is-machine-learning (Accessed: 15 December 2024).

Müller, A.C. and Guido, S. (2024) Introduction to Machine Learning with Python. Available at: https://www.nrigroupindia.com/e-book/Introduction%20to%20Machine%20Learning%20with%20Python%20(%20PDFDrive.com%20)-min.pdf (Accessed through GCU Learn: 2 December 2024).

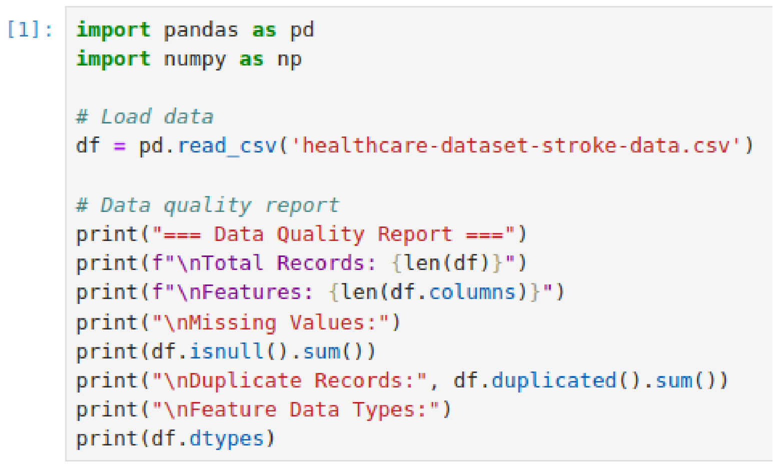
scikit-learn (2024) Random Forest Classifier. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (Accessed: 4 December 2024).

TensorFlow (2024) Feature Engineering Techniques. Available at: https://www.tensorflow.org/tutorials/structured_data/feature_columns (Accessed: 6 December 2024).

World Health Organization (2024) Stroke: Key Facts. Available at: https://www.who.int/news-room/fact-sheets/detail/stroke (Accessed: 8 December 2024).

Appendix A

A-1: Data Quality Assessment script

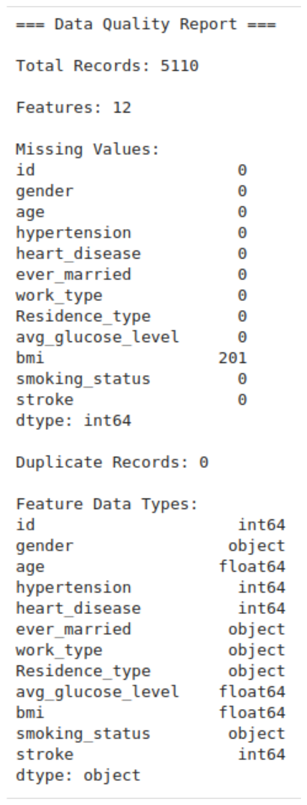


```
[1]: import pandas as pd
     import numpy as np

     # Load data
     df = pd.read_csv('healthcare-dataset-stroke-data.csv')

     # Data quality report
     print("=== Data Quality Report ===")
     print(f"\nTotal Records: {len(df)}")
     print(f"\nFeatures: {len(df.columns)}")
     print("\nMissing Values:")
     print(df.isnull().sum())
     print("\nDuplicate Records:", df.duplicated().sum())
     print("\nFeature Data Types:")
     print(df.dtypes)
```

A-2: Data Quality Assessment output



```
=== Data Quality Report ===

Total Records: 5110

Features: 12

Missing Values:
id                     0
gender                 0
age                    0
hypertension           0
heart_disease          0
ever_married           0
work_type              0
Residence_type         0
avg_glucose_level      0
bmi                  201
smoking_status         0
stroke                 0
dtype: int64

Duplicate Records: 0

Feature Data Types:
id                     int64
gender                object
age                  float64
hypertension           int64
heart_disease          int64
ever_married          object
work_type             object
Residence_type        object
avg_glucose_level    float64
bmi                  float64
smoking_status        object
stroke                 int64
dtype: object
```

[Click here to return to 1.1 Dataset Overview]

Appendix B

B-1: Feature Importance Rankings

```python
# Extract feature importances
feature_names = numerical_features.copy()
categorical_features_encoded = (
    initial_rf.named_steps['preprocessor']
    .named_transformers_['cat']
    .named_steps['onehot']
    .get_feature_names_out(categorical_features)
)
feature_names.extend(categorical_features_encoded)

importances = initial_rf.named_steps['classifier'].feature_importances_
print("\nFeature Importances:")
for name, importance in zip(feature_names, importances):
    print(f"{name}: {importance:.4f}")

# Create feature importance plot
importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
}).sort_values('Importance', ascending=True)

plt.figure(figsize=(10, 6))
plt.barh(importance_df['Feature'], importance_df['Importance'], color='skyblue')
plt.title('Feature Contribution to Stroke Prediction')
plt.xlabel('Importance')
plt.tight_layout()
plt.show()
```

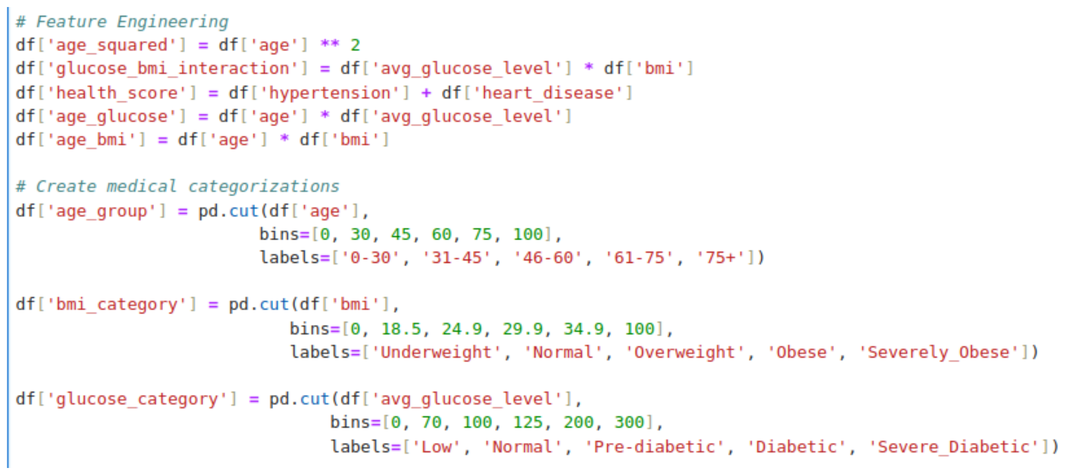
B-2: Model Implementation

```python
# Create optimized pipeline
model = ImbPipeline([
    ('preprocessor', preprocessor),
    ('smoteenn', SMOTEENN(random_state=42)),
    ('feature_selector', SelectFromModel(RandomForestClassifier(n_estimators=100, random_state=42))),
    ('classifier', RandomForestClassifier(random_state=42))
])
```

B-3: Grid Search Configuration

```python
# Optimized parameter grid
param_grid = {
     'classifier__n_estimators': [300, 400],
     'classifier__max_depth': [8, 12],
     'classifier__min_samples_split': [2, 4],
     'classifier__min_samples_leaf': [1, 2],
     'classifier__class_weight': ['balanced', 'balanced_subsample'],
     'classifier__max_features': ['sqrt', 'log2']
}
```

Appendix C: Feature Engineering



```
# Feature Engineering
df['age_squared'] = df['age'] ** 2
df['glucose_bmi_interaction'] = df['avg_glucose_level'] * df['bmi']
df['health_score'] = df['hypertension'] + df['heart_disease']
df['age_glucose'] = df['age'] * df['avg_glucose_level']
df['age_bmi'] = df['age'] * df['bmi']

# Create medical categorizations
df['age_group'] = pd.cut(df['age'],
                        bins=[0, 30, 45, 60, 75, 100],
                        labels=['0-30', '31-45', '46-60', '61-75', '75+'])

df['bmi_category'] = pd.cut(df['bmi'],
                           bins=[0, 18.5, 24.9, 29.9, 34.9, 100],
                           labels=['Underweight', 'Normal', 'Overweight', 'Obese', 'Severely_Obese'])

df['glucose_category'] = pd.cut(df['avg_glucose_level'],
                               bins=[0, 70, 100, 125, 200, 300],
                               labels=['Low', 'Normal', 'Pre-diabetic', 'Diabetic', 'Severe_Diabetic'])
```

Appendix D: Visualisation Code

D-1: Performance Visualisation

```python
# Create visualization plots
plt.figure(figsize=(15, 5))

# ROC Curve plot
plt.subplot(1, 3, 1)
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")

# Precision-Recall Curve plot
plt.subplot(1, 3, 2)
plt.plot(recall, precision, color='darkgreen', lw=2, label=f'PR curve (AUC = {pr_auc:.2f})')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend(loc="upper right")

# Confusion Matrix
plt.subplot(1, 3, 3)
cm = confusion_matrix(y_test, y_test_pred_optimal)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix (Optimal Threshold)')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')

plt.tight_layout()
plt.show()
```

D-2: Feature Distribution Visualisation

```python
# Feature importances
try:
    importances = best_model.named_steps['classifier'].feature_importances_
    feature_names = numerical_features.copy()
    categorical_features_encoded = (
        best_model.named_steps['preprocessor']
        .named_transformers_['cat']
        .named_steps['onehot']
        .get_feature_names_out(categorical_features)
    )
    feature_names.extend(categorical_features_encoded)

    # Plot feature importances
    importance_df = pd.DataFrame({
        'Feature': feature_names,
        'Importance': importances
    }).sort_values('Importance', ascending=False)

    plt.figure(figsize=(10, 6))
    sns.barplot(data=importance_df.head(10), x='Importance', y='Feature')
    plt.title('Top 10 Most Important Features')
    plt.tight_layout()
    plt.show()
except:
    print("\nCouldn't extract feature importances due to feature selection step")
```

Appendix E: Data Processing Pipeline

E-1: Preprocessing Implementation

```python
# Create preprocessing steps
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(drop='first', sparse_output=False, handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ])
```

E-2: Feature Selection

```python
initial_rf = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(
        n_estimators=300,
        max_depth=12,
        min_samples_split=2,
        min_samples_leaf=2,
        class_weight='balanced',
        max_features='log2',
        random_state=42
    ))
])

# Fit this pipeline to get feature importances
print("Calculating feature importances...")
initial_rf.fit(X_train, y_train)

# Extract feature importances
feature_names = numerical_features.copy()
categorical_features_encoded = (
    initial_rf.named_steps['preprocessor']
    .named_transformers_['cat']
    .named_steps['onehot']
    .get_feature_names_out(categorical_features)
)
feature_names.extend(categorical_features_encoded)

importances = initial_rf.named_steps['classifier'].feature_importances_
print("\nFeature Importances:")
for name, importance in zip(feature_names, importances):
    print(f"{name}: {importance:.4f}")

# Create feature importance plot
importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
}).sort_values('Importance', ascending=True)
```

Appendix F

F-1: Cross-validation

```python
# Cross-validation scores
cv_scores = cross_val_score(best_model, X_train, y_train, cv=5, scoring='f1_macro')
print("\nCross-validation F1-macro scores:", cv_scores)
print(f"Mean CV F1-macro: {cv_scores.mean():.3f} (±{cv_scores.std() * 2:.3f})")
print(f"\nOptimal prediction threshold: {best_threshold:.2f}")
```

F-2: Threshold Optimisation

```python
# Find optimal threshold
thresholds = np.arange(0.1, 0.9, 0.05)
best_threshold = 0.5
best_f1 = 0

for threshold in thresholds:
    y_pred_threshold = (y_test_prob >= threshold).astype(int)
    f1 = classification_report(y_test, y_pred_threshold, output_dict=True)['1']['f1-score']
    if f1 > best_f1:
        best_f1 = f1
        best_threshold = threshold

# Apply optimal threshold
y_test_pred_optimal = (y_test_prob >= best_threshold).astype(int)
```