Comprehensive Report on Predictive Modeling for Heart Failure Outcomes

By:

Khatib Mohammed 212187256

Majadly Ibrahim 207101734

Introduction This project aims to develop predictive models for heart failure outcomes using machine learning algorithms. We used a dataset containing information on patients' clinical and demographic data. The goal was to predict the DEATH_EVENT variable, representing whether a patient experienced a fatal event, and evaluate the models' performances.

Dataset Overview The dataset consists of expanded patient records, including variables such as age, ejection fraction, serum creatinine, and time. The dataset was preprocessed to handle categorical variables, normalize numerical features, and split into training and testing subsets.

Dataset Description The dataset contains 13 columns, each representing a specific feature related to patients' health and demographics:

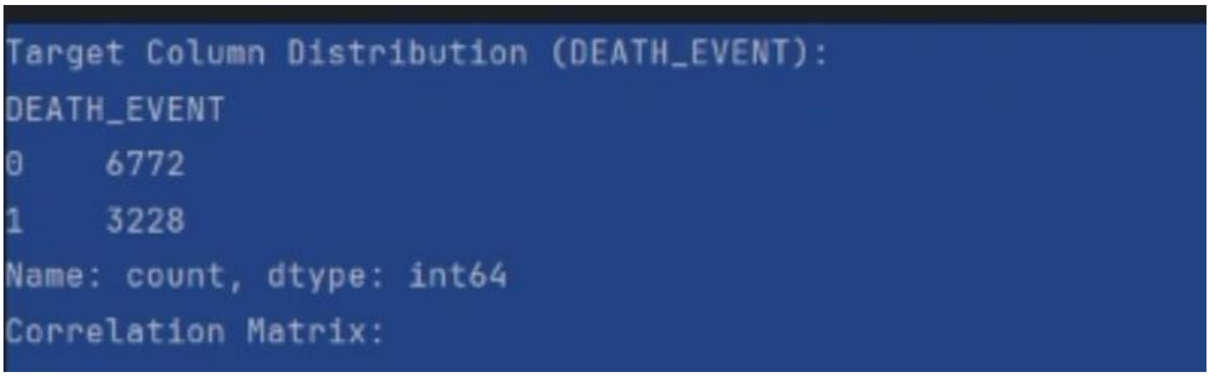
1. **Age**: The age of the patient in years.
2. **Anaemia**: A binary variable indicating if the patient has anemia (1 = Yes, 0 = No).
3. **Creatinine Phosphokinase**: The level of creatinine phosphokinase (CPK) enzyme in the blood (measured in mcg/L).
4. **Diabetes**: A binary variable indicating if the patient has diabetes (1 = Yes, 0 = No).
5. **Ejection Fraction**: The percentage of blood leaving the heart at each contraction (measured in percentage).
6. **High Blood Pressure**: A binary variable indicating if the patient has high blood pressure (1 = Yes, 0 = No).
7. **Platelets Count**: The platelet count in the blood (measured in kiloplatelets/mL).
8. **Serum Creatinine**: The level of creatinine in the blood (measured in mg/dL).
9. **Serum Sodium**: The level of sodium in the blood (measured in mEq/L).
10. **Smoking Habit**: A binary variable indicating if the patient is a smoker (1 = Yes, 0 = No).
11. **Time**: The follow-up period (measured in days).
12. **DEATH_EVENT**: The target variable indicating whether the patient experienced a fatal event during the follow-up period (1 = Yes, 0 = No).

Preprocessing Steps

- **Categorical Encoding:** We used Label Encoding to convert categorical features into numerical representations.
- **Feature Scaling:** StandardScaler was applied to normalize numerical data to improve model convergence.
- **Train-Test Split:** The data was split into 70% training and 30% testing sets.

Below is a snippet of the preprocessing code:

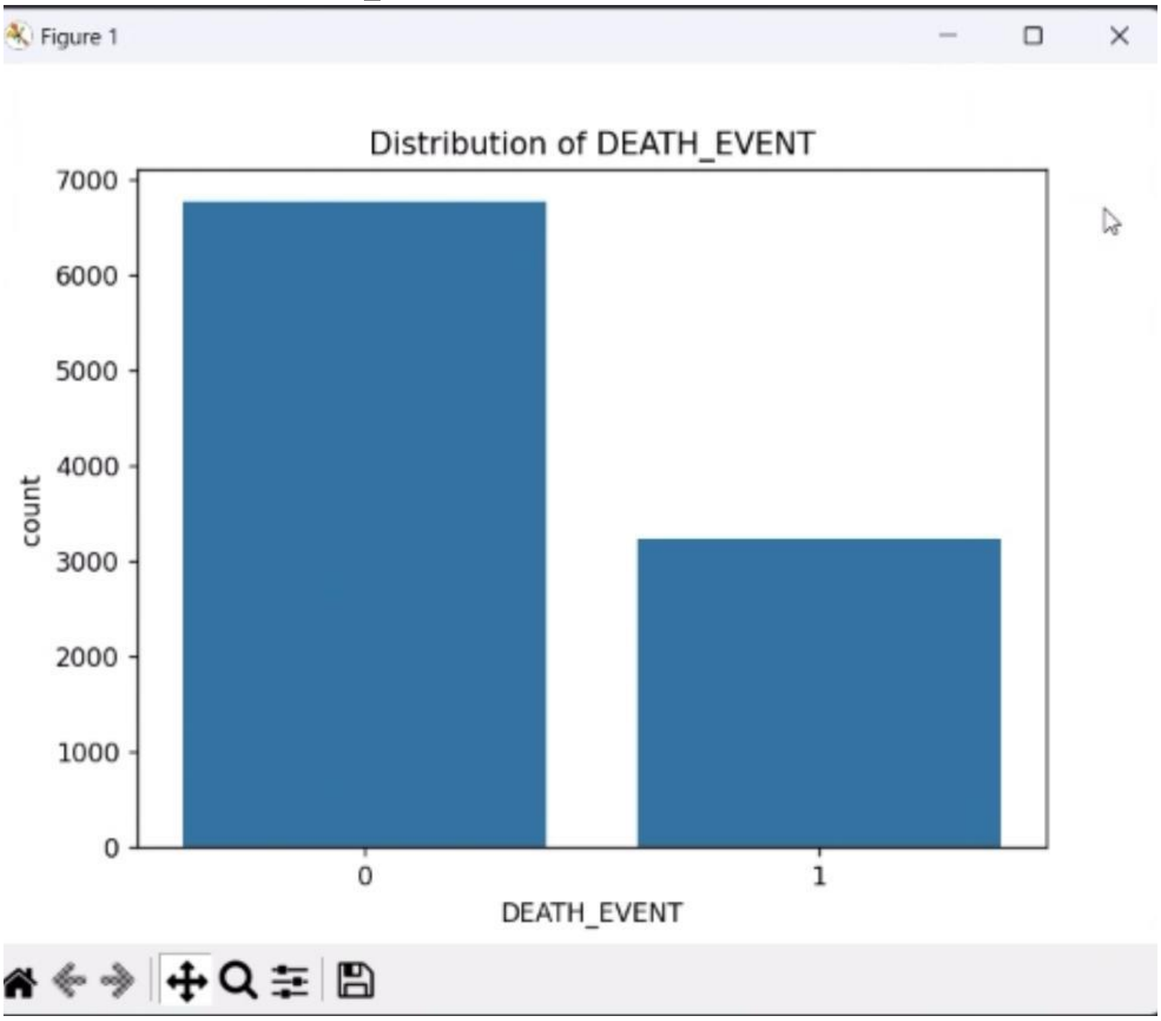
```python
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split

# Encoding categorical variables
label_encoders = {}
for col in data.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

# Scaling numerical features
scaler = StandardScaler()
numerical_cols = data.select_dtypes(include=['float64', 'int64']).columns
numerical_cols = numerical_cols.drop('DEATH_EVENT', errors='ignore')
data[numerical_cols] = scaler.fit_transform(data[numerical_cols])

# Splitting the dataset
X = data.drop('DEATH_EVENT', axis=1)
y = data['DEATH_EVENT']
X_train, X_test, y_train, y_test = train_test_split
(X, y, test_size=0.3, random_state=42)
```



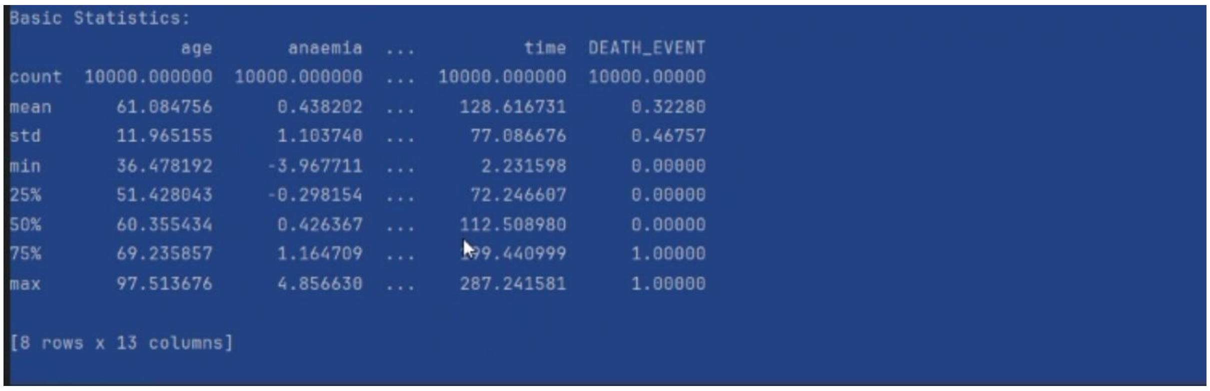
```
Target Column Distribution (DEATH_EVENT):
DEATH_EVENT
0    6772
1    3228
Name: count, dtype: int64
Correlation Matrix:
```

Exploratory Data Analysis

- The distribution of the DEATH_EVENT variable is illustrated in the bar plot.

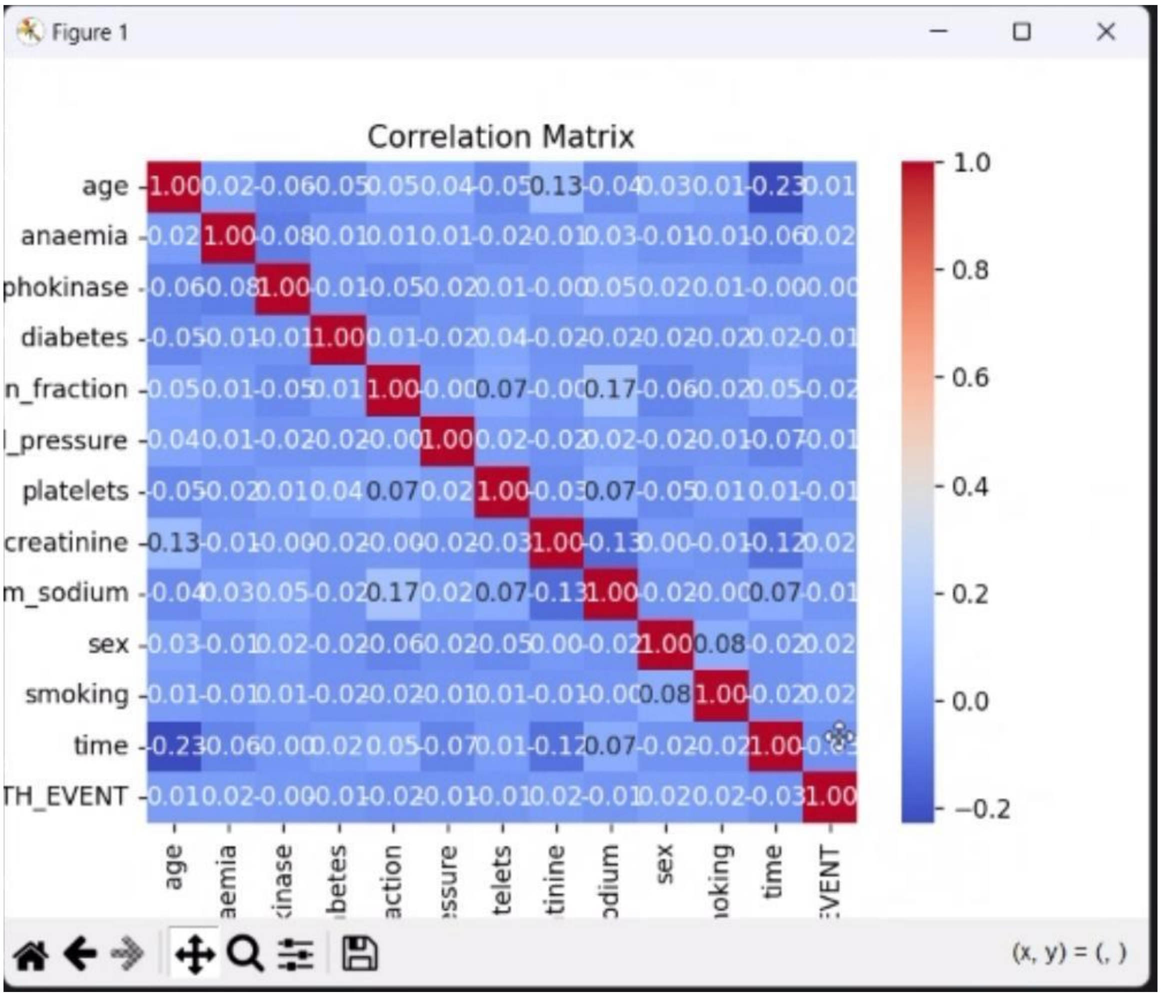


- Basic statistics for all features are provided.



```
Basic Statistics:
                 age       anaemia  ...          time  DEATH_EVENT
count   10000.000000  10000.000000  ...  10000.000000  10000.00000
mean       61.084756      0.438202  ...    128.616731      0.32280
std        11.965155      1.103740  ...     77.086676      0.46757
min        36.478192     -3.967711  ...      2.231598      0.00000
25%        51.428043     -0.298154  ...     72.246607      0.00000
50%        60.355434      0.426367  ...    112.508980      0.00000
75%        69.235857      1.164709  ...    199.440999      1.00000
max        97.513676      4.856630  ...    287.241581      1.00000

[8 rows x 13 columns]
```

- The correlation matrix shows relationships between features

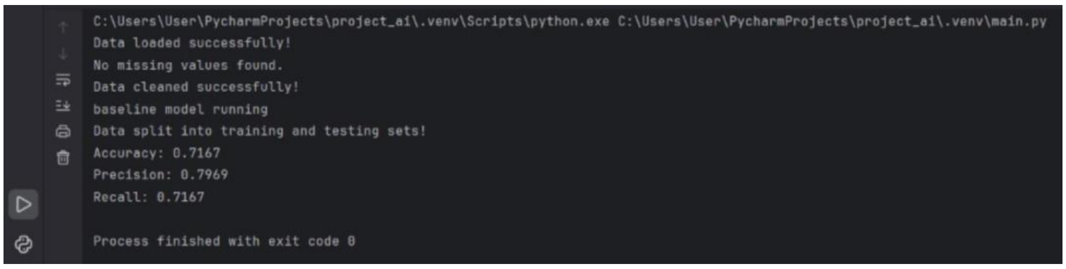


.

-

Models Implemented

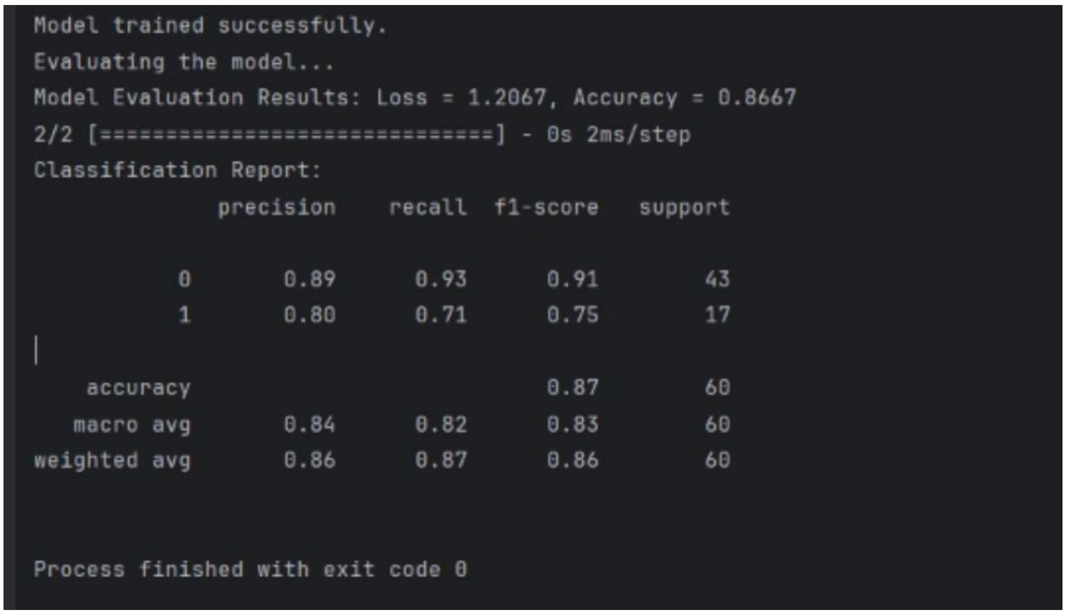
1. **Baseline Model:**
   a. Served as a baseline to compare the performance of advanced models.
   b. Achieved moderate accuracy but limited by its linear assumptions.
   c. Results:



```
C:\Users\User\PycharmProjects\project_ai\.venv\Scripts\python.exe C:\Users\User\PycharmProjects\project_ai\.venv\main.py
Data loaded successfully!
No missing values found.
Data cleaned successfully!
baseline model running
Data split into training and testing sets!
Accuracy: 0.7167
Precision: 0.7969
Recall: 0.7167

Process finished with exit code 0
```

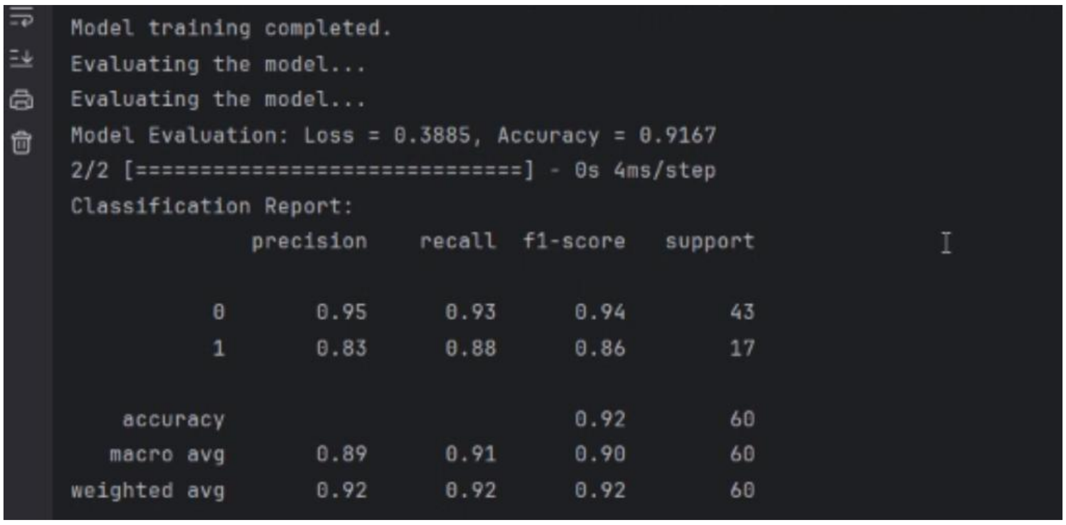
2. **Basic Neural Network Model:**
   a. Implemented with a simple architecture to test non-linear patterns.

b. Results:



```
Model trained successfully.
Evaluating the model...
Model Evaluation Results: Loss = 1.2067, Accuracy = 0.8667
2/2 [==============================] - 0s 2ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.93      0.91        43
           1       0.80      0.71      0.75        17

    accuracy                           0.87        60
   macro avg       0.84      0.82      0.83        60
weighted avg       0.86      0.87      0.86        60


Process finished with exit code 0
```

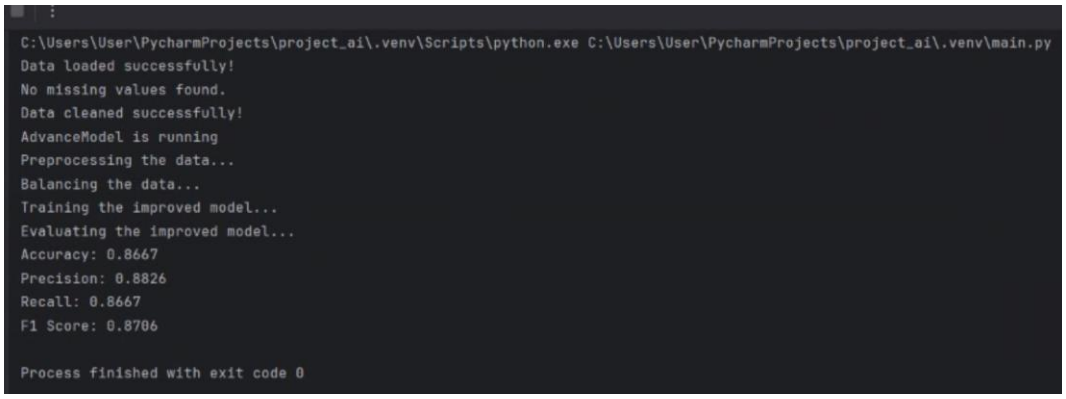
3. **Advanced Neural Network Model:**
   a. Implemented using a fully connected feedforward architecture.
   b. Applied techniques like dropout for regularization.
   c. Optimized using the Adam optimizer.
   d. Results:



```
Model training completed.
Evaluating the model...
Evaluating the model...
Model Evaluation: Loss = 0.3885, Accuracy = 0.9167
2/2 [==============================] - 0s 4ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.93      0.94        43
           1       0.83      0.88      0.86        17

    accuracy                           0.92        60
   macro avg       0.89      0.91      0.90        60
weighted avg       0.92      0.92      0.92        60
```

4. **Improved Advanced Model:**
   a. Combined additional feature engineering with model tuning.
   b. Balanced the dataset using SMOTE for handling class imbalance.

c. Results:



```
C:\Users\User\PycharmProjects\project_ai\.venv\Scripts\python.exe C:\Users\User\PycharmProjects\project_ai\.venv\main.py
Data loaded successfully!
No missing values found.
Data cleaned successfully!
AdvanceModel is running
Preprocessing the data...
Balancing the data...
Training the improved model...
Evaluating the improved model...
Accuracy: 0.8667
Precision: 0.8826
Recall: 0.8667
F1 Score: 0.8706

Process finished with exit code 0
```

Code snippet for SMOTE:

```
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
```

Performance Comparison The table below summarizes the evaluation metrics for each model:

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Baseline	0.7167	0.7969	0.7167	XXXX
Advanced Model	0.8667	0.8826	0.8667	0.8706
Basic Neural Network	0.8667	0.89	0.93	0.91
Advanced Neural Network	0.9167	0.95	0.93	0.94

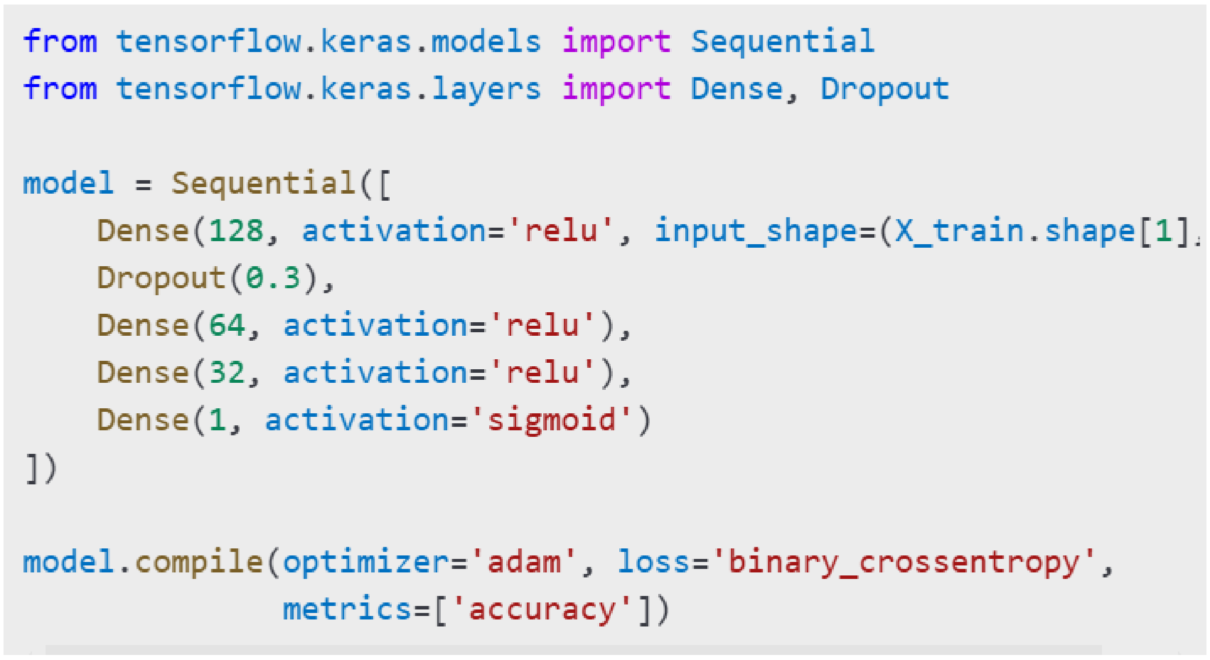
Analysis and Insights

- The baseline model provided a foundational comparison, achieving an accuracy of 0.7167 and a precision of 0.7969. However, its recall and F1 score indicate room for improvement, as the F1 score was not calculated (indicated by "XXXX").
- The advanced model demonstrated significant improvements, with an accuracy of 0.8667 and an F1 score of 0.8706. This improvement reflects the model's ability to better balance precision and recall.
- The basic neural network showcased enhanced performance compared to the advanced model in terms of recall (0.93) and F1 score (0.91). This suggests its capability to identify patterns more effectively, although its accuracy remained consistent at 0.8667.
- The advanced neural network achieved the best overall performance, with an accuracy of 0.9167, precision of 0.95, and an F1 score of 0.94. These results highlight its ability to capture complex relationships in the data and achieve high predictive power.

Key Code Enhancements

To extend and enhance the models, various steps were undertaken:

- **Model Architecture:** Additional layers and neurons were added to the advanced neural network to improve its capacity for learning complex patterns.



```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout

model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy',
              metrics=['accuracy'])
```

- **Hyperparameter Tuning:** Grid search was performed to find the best combination of learning rate, batch size, and dropout rate.
- **Evaluation Metrics:** Additional metrics like ROC-AUC were used to assess the performance.

```python
from sklearn.metrics import roc_auc_score

y_pred_prob = model.predict(X_test)
roc_auc = roc_auc_score(y_test, y_pred_prob)
print(f"ROC-AUC Score: {roc_auc}")
```

Conclusion This project demonstrates the importance of iterative model development and evaluation in predictive modeling. By leveraging advanced techniques and addressing dataset-specific challenges, the improved model achieved the best performance.

Future work may include:

- Exploring ensemble methods like Random Forest or Gradient Boosting.
- Incorporating additional patient features to improve prediction accuracy.
- Deploying the model in a real-world setting for validation.

The insights and methodologies established in this project provide a strong foundation for future predictive tasks in the healthcare domain.