In [1]:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_auc_score

from sklearn.naive_bayes import GaussianNB

from sklearn.model_selection import KFold
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.callbacks import EarlyStopping
```

In [2]:

```
data = pd.read_csv("/content/heart_failure_clinical_records_dataset.csv")
```

In [3]:

```
data.describe()
```

Out[3]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_c
count	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	2
mean	60.833893	0.431438	581.839465	0.418060	38.083612	0.351171	263358.029264	
std	11.894809	0.496107	970.287881	0.494067	11.834841	0.478136	97804.236869	
min	40.000000	0.000000	23.000000	0.000000	14.000000	0.000000	25100.000000	
25%	51.000000	0.000000	116.500000	0.000000	30.000000	0.000000	212500.000000	
50%	60.000000	0.000000	250.000000	0.000000	38.000000	0.000000	262000.000000	
75%	70.000000	1.000000	582.000000	1.000000	45.000000	1.000000	303500.000000	
max	95.000000	1.000000	7861.000000	1.000000	80.000000	1.000000	850000.000000	

In [ ]:

```
data.shape
```

Out[ ]: (299, 13)

In [4]:

```
column_names = list(data.columns)

print(column_names)
```

```
['age', 'anaemia', 'creatinine_phosphokinase', 'diabetes', 'ejection_fraction', 'high_blood_pressure', 'platelet
s', 'serum_creatinine', 'serum_sodium', 'sex', 'smoking', 'time', 'DEATH_EVENT']
```

In [21]:

```
X = data.drop(['sex', 'time', 'DEATH_EVENT'], axis=1)
```

In [22]:

```
y = data['DEATH_EVENT']
```

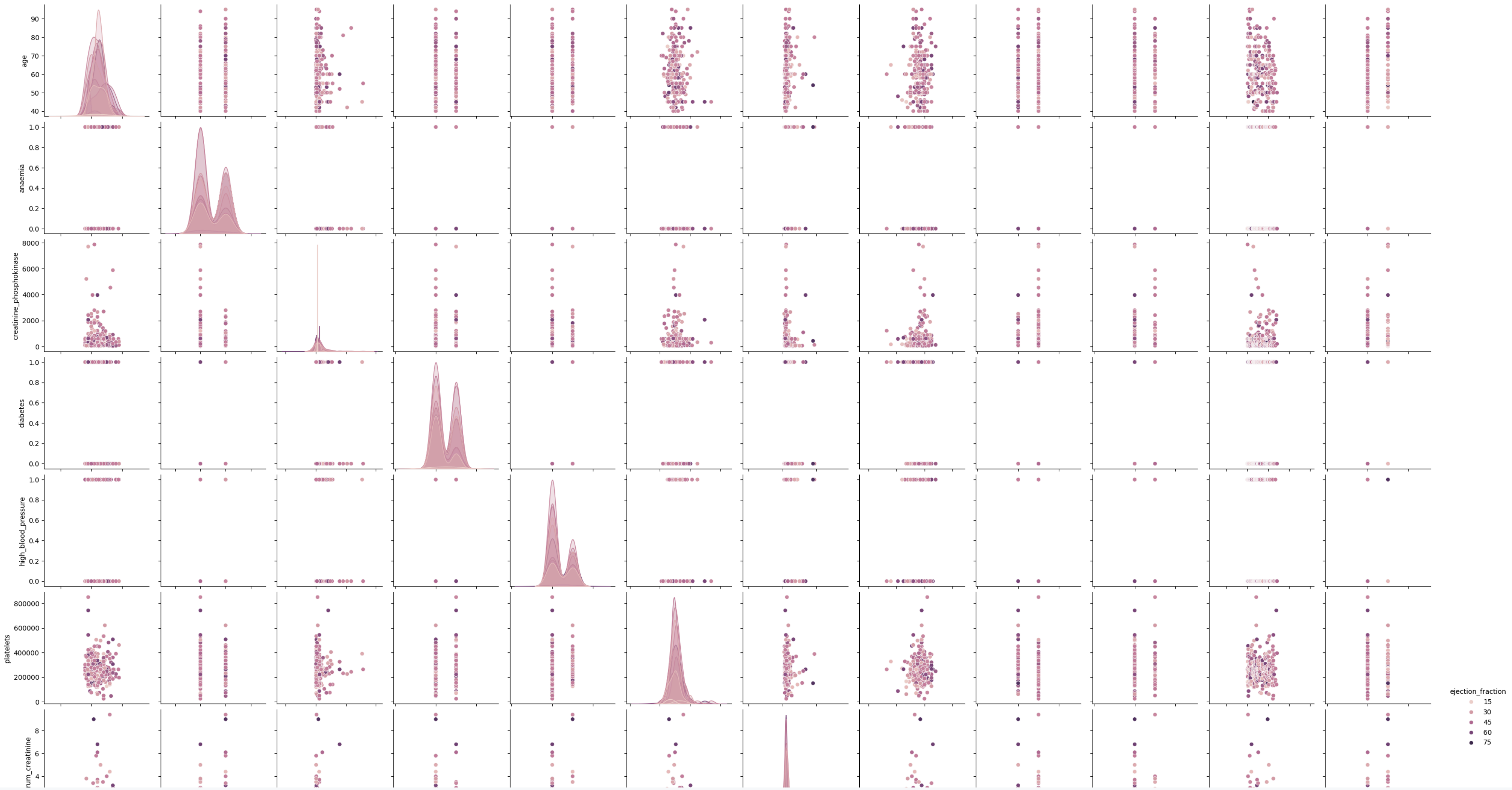
In [23]:

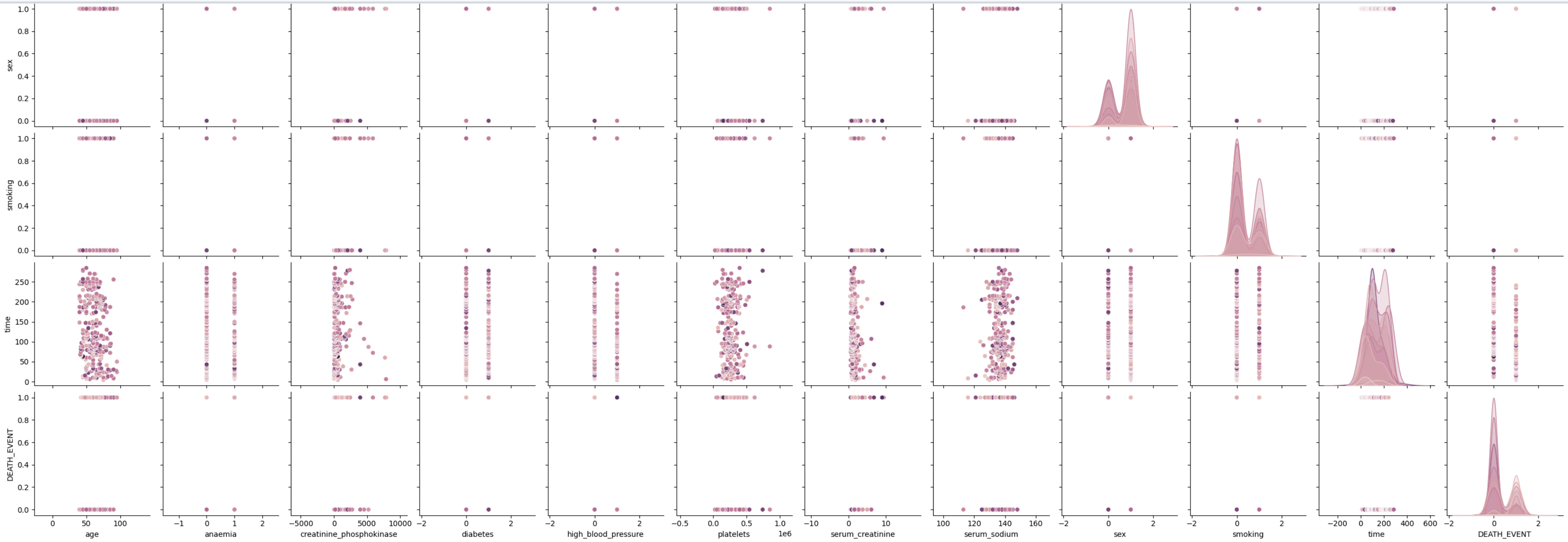
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature scaling
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [30]:

```
## Exploratory Data Analysis (EDA)
# the distribution of numerical features and their relationship with the target variable
sns.pairplot(data, hue="ejection_fraction", diag_kind="kde")
plt.show()
```





In [25]:

```
# Naive Bayes classifier
nb_classifier = GaussianNB()
nb_classifier.fit(X_train_scaled, y_train)
nb_pred = nb_classifier.predict(X_test_scaled)
nb_accuracy = accuracy_score(y_test, nb_pred)


# Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train_scaled, y_train)
rf_pred = rf_classifier.predict(X_test_scaled)
rf_accuracy = accuracy_score(y_test, rf_pred)
```

In [26]:

```
# TensorFlow model
model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
```

```
    Dense(1, activation='sigmoid')
])


model.compile(optimizer='adam', loss='mean_squared_error', metrics=['mae'])  # MSE for regression
```

In [27]:

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

history = model.fit(X_train, y_train, epochs=1000, batch_size=32, validation_split=0.2, verbose=0)

# Evaluate on the test set
test_loss, test_mae = model.evaluate(X_test_scaled, y_test, verbose=0)

# Print the results
print("\n--- Evaluation Results ---")
print(f"Naive Bayes Accuracy: {nb_accuracy:.4f}")
print(f"Random Forest Accuracy: {rf_accuracy:.4f}")
print(f"TensorFlow Model Test Loss (MSE): {test_loss:.4f}")
print(f"TensorFlow Model Test Mean Absolute Error (MAE): {test_mae:.4f}")
```

```
--- Evaluation Results ---
Naive Bayes Accuracy: 0.6333
Random Forest Accuracy: 0.7000
TensorFlow Model Test Loss (MSE): 0.7277
TensorFlow Model Test Mean Absolute Error (MAE): 0.6333
```

In [29]:

```
# Plot training history
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
```

Out[29]: <matplotlib.legend.Legend at 0x7fc9e415f580>