```
!pip install torch torchvision torchaudio
!pip install transformers
```

```
  Found existing installation: torch 2.2.1
  Uninstalling torch-2.2.1:
    Would remove:
      /usr/local/bin/convert-caffe2-to-onnx
      /usr/local/bin/convert-onnx-to-caffe2
      /usr/local/bin/torchrun
      /usr/local/lib/python3.10/dist-packages/functorch/*
      /usr/local/lib/python3.10/dist-packages/torch-2.2.1.dist-info/*
      /usr/local/lib/python3.10/dist-packages/torch/*
      /usr/local/lib/python3.10/dist-packages/torchgen/*
  Proceed (Y/n)? Y
    Successfully uninstalled torch-2.2.1
  WARNING: Skipping transformers as it is not installed.
  Collecting torch
  Y
    Using cached torch-2.2.2-cp310-cp310-manylinux1_x86_64.whl (755.5 MB)
  Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-packages (0.17.1+cu121)
  Requirement already satisfied: torchaudio in /usr/local/lib/python3.10/dist-packages (2.2.1+cu121)
  Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.13.4)
  Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch) (4.11.0)
  Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch) (1.12)
  Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.3)
  Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.3)
  Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch) (2023.6.0)
  Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch) (12.1.105)
  Requirement already satisfied: nvidia-cuda-runtime-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch) (12.1.105)
  Requirement already satisfied: nvidia-cuda-cupti-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch) (12.1.105)
  Requirement already satisfied: nvidia-cudnn-cu12==8.9.2.26 in /usr/local/lib/python3.10/dist-packages (from torch) (8.9.2.26)
  Requirement already satisfied: nvidia-cublas-cu12==12.1.3.1 in /usr/local/lib/python3.10/dist-packages (from torch) (12.1.3.1)
  Requirement already satisfied: nvidia-cufft-cu12==11.0.2.54 in /usr/local/lib/python3.10/dist-packages (from torch) (11.0.2.54)
  Requirement already satisfied: nvidia-curand-cu12==10.3.2.106 in /usr/local/lib/python3.10/dist-packages (from torch) (10.3.2.106)
  Requirement already satisfied: nvidia-cusolver-cu12==11.4.5.107 in /usr/local/lib/python3.10/dist-packages (from torch) (11.4.5.107)
  Requirement already satisfied: nvidia-cusparse-cu12==12.1.0.106 in /usr/local/lib/python3.10/dist-packages (from torch) (12.1.0.106)
  Requirement already satisfied: nvidia-nccl-cu12==2.19.3 in /usr/local/lib/python3.10/dist-packages (from torch) (2.19.3)
  Requirement already satisfied: nvidia-nvtx-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch) (12.1.105)
  Requirement already satisfied: triton==2.2.0 in /usr/local/lib/python3.10/dist-packages (from torch) (2.2.0)
  Requirement already satisfied: nvidia-nvjitlink-cu12 in /usr/local/lib/python3.10/dist-packages (from nvidia-cusolver-cu12==11.4.5.107
  Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from torchvision) (1.25.2)
    Using cached torch-2.2.1-cp310-cp310-manylinux1_x86_64.whl (755.5 MB)
  Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.10/dist-packages (from torchvision) (9.4.0)
  Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch) (2.1.5)
  Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch) (1.3.0)
  Installing collected packages: torch
  Successfully installed torch-2.2.1
  Collecting transformers
    Using cached transformers-4.40.0-py3-none-any.whl (9.0 MB)
  Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.13.4)
  Requirement already satisfied: huggingface-hub<1.0,>=0.19.3 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.20.3)
  Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.25.2)
  Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (24.0)
  Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.1)
  Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2023.12.25)
  Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.31.0)
  Collecting tokenizers<0.20,>=0.19 (from transformers)
    Downloading tokenizers-0.19.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.6 MB)
       ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.6/3.6 MB 33.0 MB/s eta 0:00:00
  Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.4.2)
```

```
pip install optuna
```

```
  Requirement already satisfied: optuna in /usr/local/lib/python3.10/dist-packages (3.6.1)
  Requirement already satisfied: alembic>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from optuna) (1.13.1)
  Requirement already satisfied: colorlog in /usr/local/lib/python3.10/dist-packages (from optuna) (6.8.2)
  Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from optuna) (1.25.2)
  Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from optuna) (24.0)
  Requirement already satisfied: sqlalchemy>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from optuna) (2.0.29)
  Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from optuna) (4.66.2)
  Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-packages (from optuna) (6.0.1)
  Requirement already satisfied: Mako in /usr/local/lib/python3.10/dist-packages (from alembic>=1.5.0->optuna) (1.3.3)
  Requirement already satisfied: typing-extensions>=4 in /usr/local/lib/python3.10/dist-packages (from alembic>=1.5.0->optuna) (4.11.0)
  Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.10/dist-packages (from sqlalchemy>=1.3.0->optuna) (3.0.3)
  Requirement already satisfied: MarkupSafe>=0.9.2 in /usr/local/lib/python3.10/dist-packages (from Mako->alembic>=1.5.0->optuna) (2.1.5)
```

```
from google.colab import drive
import pandas as pd
import numpy as np
import torch
from transformers import BertTokenizer, BertForSequenceClassification, AdamW
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
import optuna
from joblib import dump
import warnings
import re

# Ignore warnings
warnings.filterwarnings('ignore')

# Define global variables for X_train, y_train, X_val, and y_val
X_train, y_train, X_val, y_val = None, None, None, None

def load_data():
    # Mount Google Drive
    drive.mount('/content/drive')

    # Load the DataFrames
    folder_path = '/content/drive/My Drive/Hs/'

    # Load the datasets
    admissions = pd.read_csv(folder_path + 'admissions.csv')
    patients = pd.read_csv(folder_path + 'patients.csv')
    icustays = pd.read_csv(folder_path + 'icustays.csv')
    diagnoses_icd = pd.read_csv(folder_path + 'diagnoses_icd.csv')
    d_icd_diagnoses = pd.read_csv(folder_path + 'd_icd_diagnoses.csv')

    # Merge datasets based on common columns using inner join
    merged_data = pd.merge(admissions, patients, on='subject_id', how='inner')
    merged_data = pd.merge(merged_data, icustays, on=['subject_id', 'hadm_id'], how='inner')
    merged_data = pd.merge(merged_data, diagnoses_icd, on=['subject_id', 'hadm_id'], how='inner')
    d_icd_diagnoses['long_title'] = d_icd_diagnoses['long_title'].apply(preprocess_text)
    merged_data = pd.merge(merged_data, d_icd_diagnoses, on=['icd_code', 'icd_version'], how='inner')
    merged_data.rename(columns={'hospital_expire_flag': 'label', 'long_title': 'text'}, inplace=True)

    # Feature Engineering
    categories = ['Short', 'Medium', 'Long']
    short_threshold = 2.0
    medium_threshold = 5.0

    # Categorize Length of Stay (LOS)
    merged_data['LOS_Category'] = pd.cut(merged_data['los'], bins=[0, short_threshold, medium_threshold, float('inf')],
                                         labels=categories, right=False)

    # Drop unnecessary columns
    columns_to_drop = ['deathtime', 'admit_provider_id', 'dod', 'los', 'language', 'seq_num', 'insurance']
    merged_data = merged_data.drop(columns=columns_to_drop)

    # Handling missing values
    merged_data.dropna(inplace=True)

    return merged_data

def preprocess_text(text):
    # Remove punctuation using regex
    text = re.sub(r'[^\w\s]', '', text)
    return text

# Define objective function for Optuna
def objective(trial):
    # Define parameters to search
    lr = trial.suggest_loguniform('lr', 1e-6, 1e-4)
    epochs = trial.suggest_int('epochs', 3, 5)
    batch_size = trial.suggest_categorical('batch_size', [16, 32, 64])

    # Initialize BERT tokenizer and model
    tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
    model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=3)  # Adjust for 3 classes

    # Define optimizer
    optimizer = AdamW(model.parameters(), lr=lr)
```

```
    # Split data into train, validation, and test sets
    X_train_val, X_test, y_train_val, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
    X_train, X_val, y_train, y_val = train_test_split(X_train_val, y_train_val, test_size=0.111, random_state=42)  # 80% train, 10% validati

    # Tokenize input data
    train_encodings = tokenizer(X_train["text"].tolist(), padding=True, truncation=True, return_tensors='pt')
    val_encodings = tokenizer(X_val["text"].tolist(), padding=True, truncation=True, return_tensors='pt')

    # Convert labels to tensors
    train_labels = torch.tensor(y_train.tolist())
    val_labels = torch.tensor(y_val.tolist())

    # Train the model
    for epoch in range(epochs):
        model.train()
        optimizer.zero_grad()

        # Forward pass
        train_outputs = model(**train_encodings, labels=train_labels)
        train_loss = train_outputs.loss

        # Backward pass
        train_loss.backward()
        optimizer.step()

        # Evaluation
        model.eval()
        with torch.no_grad():
            val_outputs = model(**val_encodings)
            val_predictions = val_outputs.logits.argmax(dim=1)
            val_accuracy = accuracy_score(val_labels, val_predictions)

    return val_accuracy

# Load data
merged_df = load_data()

# Split data into features (X) and target (y)
X = merged_df.drop(columns=["label"])
y = merged_df["label"]

# Run hyperparameter optimization
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=100)

# Print best parameters and performance
print("Best parameters found: ", study.best_params)
print("Best accuracy on validation set: {:.4f}".format(study.best_value))

# Get the best hyperparameters
best_params = study.best_params

# Reinitialize the model with the best hyperparameters
best_tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
best_model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=3)  # Adjust for 3 classes

# Define optimizer with best learning rate
best_optimizer = AdamW(best_model.parameters(), lr=best_params['lr'])

# Train the best model
for epoch in range(best_params['epochs']):
    best_model.train()
    best_optimizer.zero_grad()

    # Forward pass
    train_encodings = best_tokenizer(X_train["text"].tolist(), padding=True, truncation=True, return_tensors='pt')
    train_labels = torch.tensor(y_train.tolist())
    train_outputs = best_model(**train_encodings, labels=train_labels)
    train_loss = train_outputs.loss

    # Backward pass
    train_loss.backward()
    best_optimizer.step()

# Save the best model
model_filename = "BERT_Model.pkl"
torch.save(best_model.state_dict(), model_filename)
```

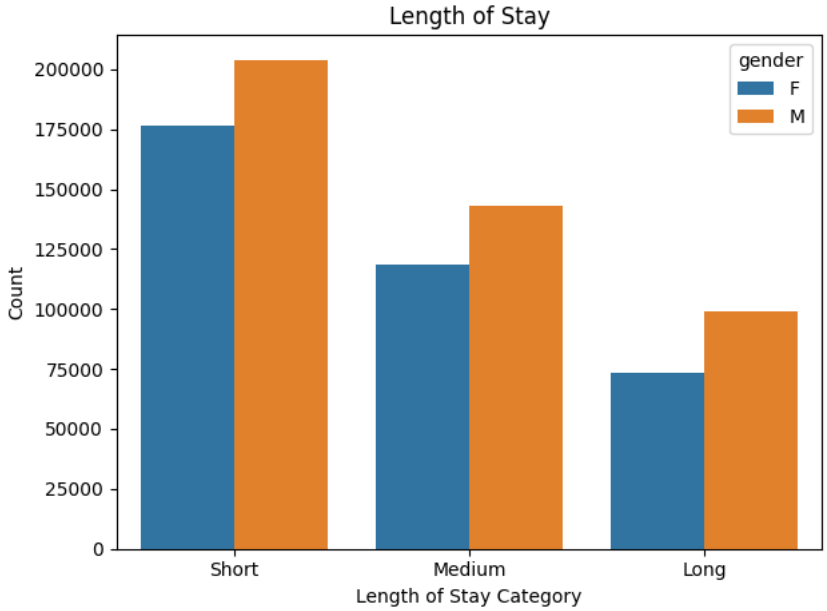
```
print(f"Best model saved to {model_filename}")

# Evaluate the best model on the test set
test_encodings = best_tokenizer(X_test["text"].tolist(), padding=True, truncation=True, return_tensors='pt')
test_labels = torch.tensor(y_test.tolist())
best_model.eval()
with torch.no_grad():
    test_outputs = best_model(**test_encodings)
    test_predictions = test_outputs.logits.argmax(dim=1)
    print(classification_report(test_labels.numpy(), test_predictions.numpy()))
```

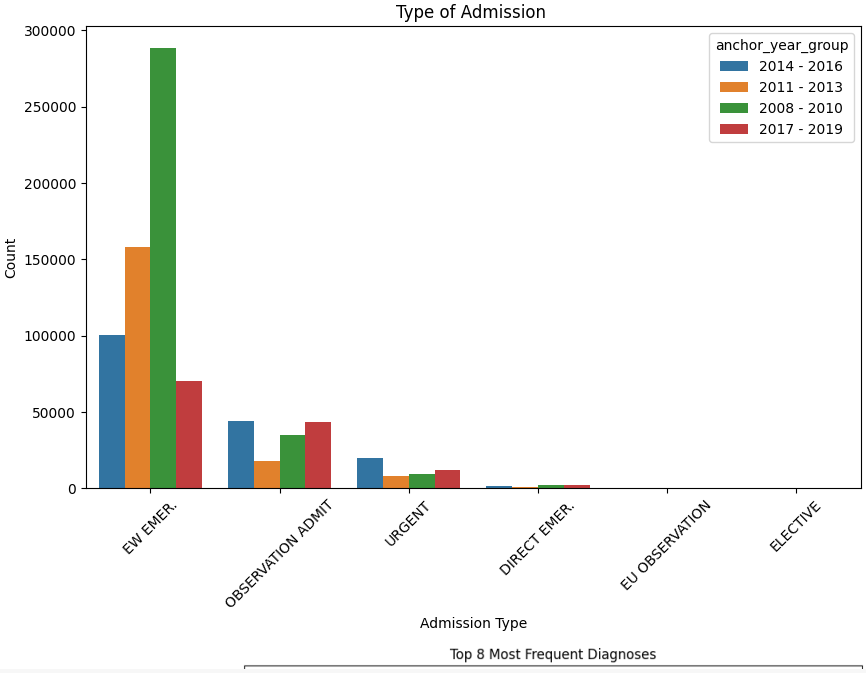
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
[I 2024-04-18 17:08:50,912] A new study created in memory with name: no-name-8a2f06e7-d2b6-4a12-8fb0-89daf99f893e
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly initiali
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```
# Custom BERT Classifier
class ClinicaBERTClassifier(BaseEstimator, ClassifierMixin):
    def __init__(self, model_name='emilyalsentzer/Bio_ClinicalBERT', lr=2e-5, epochs=3, batch_size=32, early_stopping=True, patience=3):
        self.model_name = model_name
        self.lr = lr
        self.epochs = epochs
        self.batch_size = batch_size
        self.early_stopping = early_stopping
        self.patience = patience

    def fit(self, X, y):
        self.model = BertForSequenceClassification.from_pretrained(self.model_name, num_labels=2)
        self.tokenizer = BertTokenizer.from_pretrained(self.model_name)
        self.optimizer = AdamW(self.model.parameters(), lr=self.lr)
        best_val_loss = float('inf')
        patience_counter = 0

        # Tokenize text data and prepare input tensors
        inputs = self.tokenizer(X["text"].tolist(), padding=True, truncation=True, return_tensors='pt', max_length=128)
        labels = torch.tensor(y.tolist())

        # Train the model
        for epoch in range(self.epochs):
            self.model.train()
            running_loss = 0.0
            for i in range(0, len(labels), self.batch_size):
                input_batch = {k: v[i:i+self.batch_size] for k, v in inputs.items()}
                label_batch = labels[i:i+self.batch_size]

                self.optimizer.zero_grad()
                outputs = self.model(**input_batch, labels=label_batch)
                loss = outputs.loss
                loss.backward()
                self.optimizer.step()

                running_loss += loss.item()

            # Early stopping
            if self.early_stopping:
                val_loss = self.evaluate_loss(X_val, y_val)
                if val_loss < best_val_loss:
                    best_val_loss = val_loss
                    patience_counter = 0
                else:
                    patience_counter += 1
                    if patience_counter >= self.patience:
                        print("Early stopping after {} epochs.".format(epoch + 1))
                        break

    def predict(self, X):
        # Tokenize text data and prepare input tensors
        inputs = self.tokenizer(X["text"].tolist(), padding=True, truncation=True, return_tensors='pt', max_length=128)

        # Evaluate the model
        self.model.eval()
        predictions = []
        with torch.no_grad():
            for i in range(0, len(inputs['input_ids']), self.batch_size):
                input_batch = {k: v[i:i+self.batch_size] for k, v in inputs.items()}
                outputs = self.model(**input_batch)
                logits = outputs.logits
                predictions.extend(logits.argmax(dim=1).cpu().tolist())
        return predictions

    def evaluate_loss(self, X, y):
        inputs = self.tokenizer(X["text"].tolist(), padding=True, truncation=True, return_tensors='pt', max_length=128)
        labels = torch.tensor(y.tolist())

        self.model.eval()
        total_loss = 0.0
        with torch.no_grad():
            for i in range(0, len(labels), self.batch_size):
                input_batch = {k: v[i:i+self.batch_size] for k, v in inputs.items()}
                label_batch = labels[i:i+self.batch_size]
                outputs = self.model(**input_batch, labels=label_batch)
```

```
                loss = outputs.loss
                total_loss += loss.item()

        return total_loss / (len(labels) / self.batch_size)

# Define objective function for Optuna
def objective(trial):
    # Define parameters to search
    lr = trial.suggest_loguniform('lr', 1e-6, 1e-4)
    epochs = trial.suggest_int('epochs', 3, 5)
    batch_size = trial.suggest_categorical('batch_size', [16, 32, 64])

    # Initialize ClinicaBERT Classifier with dynamically passed values
    clinica_bert_classifier = ClinicaBERTClassifier(
        model_name='emilyalsentzer/Bio_ClinicalBERT',
        lr=lr,
        epochs=epochs,
        batch_size=batch_size
    )

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

    # Evaluate using validation set
    predictions = clinica_bert_classifier.predict(X_val)
    accuracy = accuracy_score(y_val, predictions)
    return accuracy
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mo
```

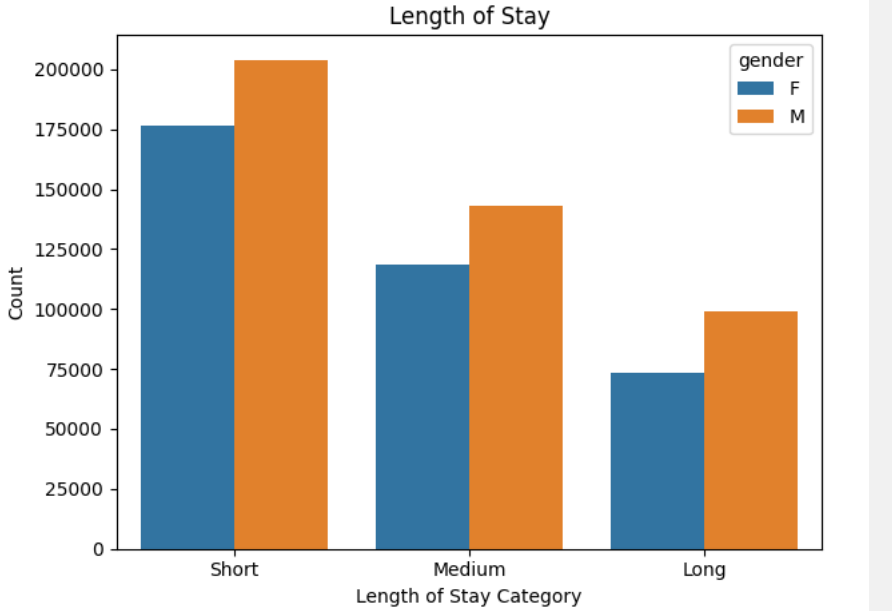


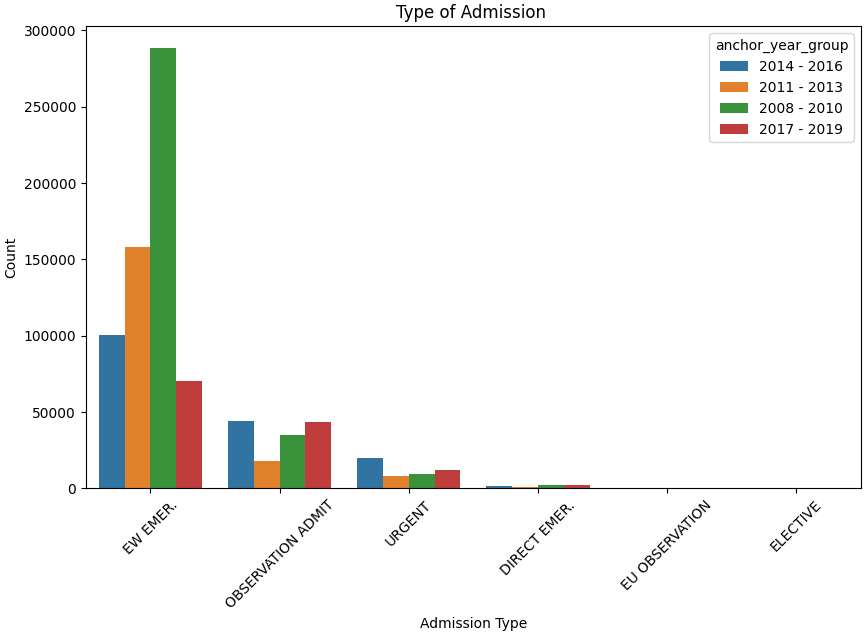


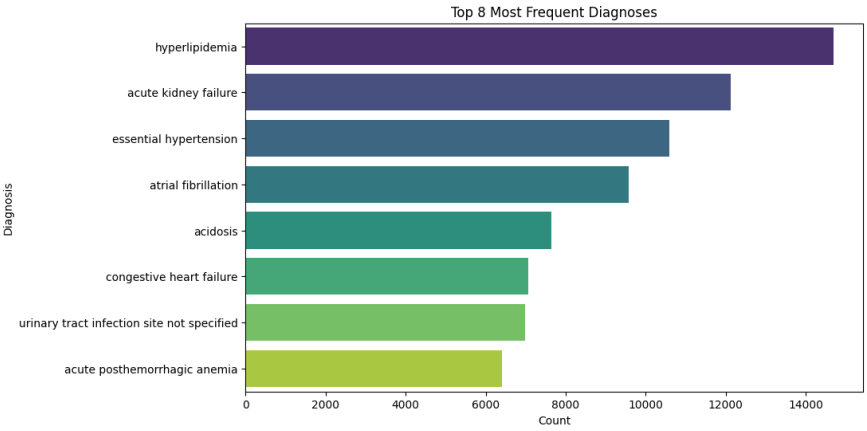
```
import re
from google.colab import drive
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder

from sklearn.base import BaseEstimator, ClassifierMixin
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from joblib import dump
import warnings
import optuna
import torch
from transformers import BertTokenizer, BertForSequenceClassification, AdamW

# Ignore warnings
warnings.filterwarnings('ignore')

def load_data():

    # Mount Google Drive
    drive.mount('/content/drive')

    # Load the DataFrames
    folder_path = '/content/drive/My Drive/Hs/'
    admissions = pd.read_csv(folder_path + 'admissions.csv')
    patients = pd.read_csv(folder_path + 'patients.csv')
    icustays = pd.read_csv(folder_path + 'icustays.csv')
    diagnoses_icd = pd.read_csv(folder_path + 'diagnoses_icd.csv')
    d_icd_diagnoses = pd.read_csv(folder_path + 'd_icd_diagnoses.csv')

    # Merge datasets based on common columns using inner join
    merged_data = pd.merge(admissions, patients, on='subject_id', how='inner')
    merged_data = pd.merge(merged_data, icustays, on=['subject_id', 'hadm_id'], how='inner')
    merged_data = pd.merge(merged_data, diagnoses_icd, on=['subject_id', 'hadm_id'], how='inner')
    d_icd_diagnoses['long_title'] = d_icd_diagnoses['long_title'].apply(preprocess_text)
    merged_data = pd.merge(merged_data, d_icd_diagnoses, on=['icd_code', 'icd_version'], how='inner')
    merged_data.rename(columns={'hospital_expire_flag': 'label', 'long_title': 'text'}, inplace=True)

    # Feature Engineering
    categories = ['Short', 'Medium', 'Long']
    short_threshold = 2.0
    medium_threshold = 5.0

    # Categorize Length of Stay (LOS)
    merged_data['LOS_Category'] = pd.cut(merged_data['los'], bins=[0, short_threshold, medium_threshold, float('inf')],
                                         labels=categories, right=False)

    # Drop unnecessary columns
    columns_to_drop = ['deathtime', 'admit_provider_id', 'dod', 'los', 'language', 'seq_num', 'insurance']
    merged_data = merged_data.drop(columns=columns_to_drop)

    # Handling missing values
    merged_data.dropna(inplace=True)

    return merged_data


def preprocess_text(text):

    # Define words to remove (converted to lowercase)
    words_to_remove = ['and', 'or', 'unspecified', 'other']
    # Remove punctuation using regex
    text = re.sub(r'[^\w\s,]', '', text)
    # Remove commas
    text = text.replace(',', '')
    # Remove specified words (case insensitive)
    text = ' '.join(word for word in text.split() if word.lower() not in words_to_remove)
    return text


def create_data_analysis_charts(data):

    # Plot countplot
    ax = sns.countplot(data=data, x='LOS_Category', hue='gender')
```

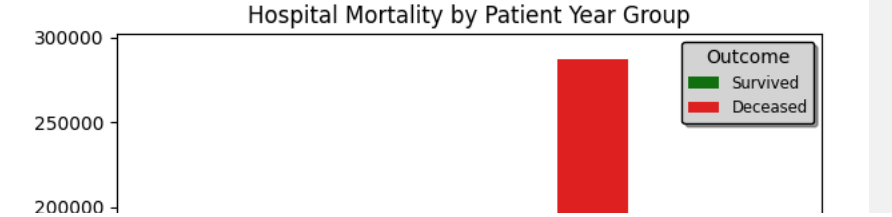
```
    ax.set_title('Length of Stay')
    ax.set_xlabel('Length of Stay Category')
    ax.set_ylabel('Count')
    # Adjust layout and display the plot
    plt.tight_layout()
    plt.show()

    # Plot for the top 6 admission types
    plt.figure(figsize=(10, 6))
    sns.countplot(data=data, x='admission_type', order=data['admission_type'].value_counts().index[:6],  hue='anchor_year_group')
    plt.title('Type of Admission ')
    plt.xlabel('Admission Type')
    plt.ylabel('Count')
    plt.xticks(rotation=45)
    plt.show()

    top_n = 8
    # Convert all diagnoses to lowercase to ensure case-insensitive comparison
    data['text'] = data['text'].str.lower()
    # Get the top N most frequent diagnoses after removing duplicates
    top_diagnoses = data['text'].value_counts().nlargest(top_n)
    # Plot the horizontal bar chart
    plt.figure(figsize=(10, 6))
    sns.barplot(x=top_diagnoses.values, y=top_diagnoses.index, palette='viridis')
    plt.title(f'Top {top_n} Most Frequent Diagnoses')
    plt.xlabel('Count')
    plt.ylabel('Diagnosis')
    plt.show()

    # Map original label values to custom labels
    label_mapping = {1: 'Deceased', 0: 'Survived'}
    # Set custom colors for each label
    custom_palette = {1: 'green', 0: 'red'}
    # Plot countplot
    ax = sns.countplot(data=data, x='anchor_year_group', hue='label', palette=custom_palette, hue_order=[1, 0])
    ax.set_title('Hospital Mortality by Patient Year Group')
    ax.set_xlabel('Patient Year Group')
    ax.set_ylabel('Count')
    # Set custom labels for hue variable
    legend_labels = [label_mapping[label] for label in sorted(data['label'].unique())]
    # Set custom colors for legend
    legend_colors = [custom_palette[label] for label in sorted(data['label'].unique())]
    ax.legend(title='Outcome', labels=legend_labels, loc='upper right', fontsize='small', facecolor='lightgrey', edgecolor='black', fancybox
    # Adjust layout and display the plot
    plt.tight_layout()
    plt.show()

    # Encode categorical variables
    encoder = LabelEncoder()
    categorical_columns = ['admission_type','icd_code', 'marital_status', 'ETHNICITY', 'LOS_Category', 'gender', 'anchor_year_group','label'
    plt.figure(figsize=(10, 8))
    # Exclude non-numeric columns from the correlation matrix
    numeric_data = data.select_dtypes(include=[np.number])
    for col in categorical_columns:
        numeric_data[col] = encoder.fit_transform(data[col])
    sns.heatmap(numeric_data.corr(), annot=True, cmap='coolwarm', fmt=".2f")
    plt.title('Correlation Matrix')
    plt.show()


# Load data
merged_df = load_data()
# Create data analysis charts
create_data_analysis_charts(merged_df)
```

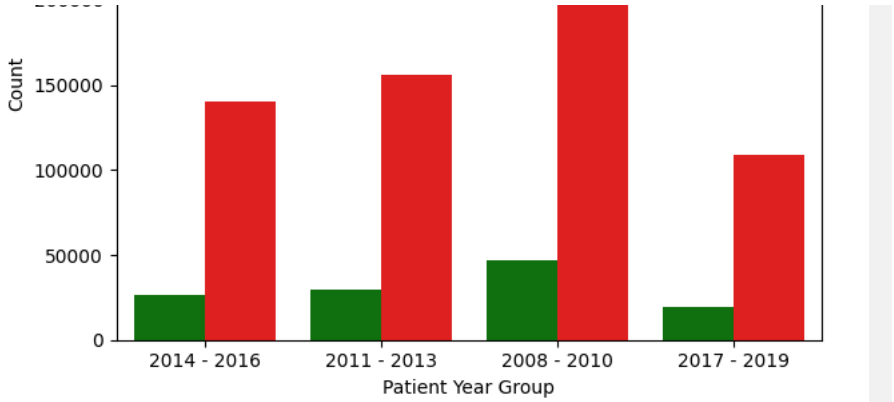
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mo
```

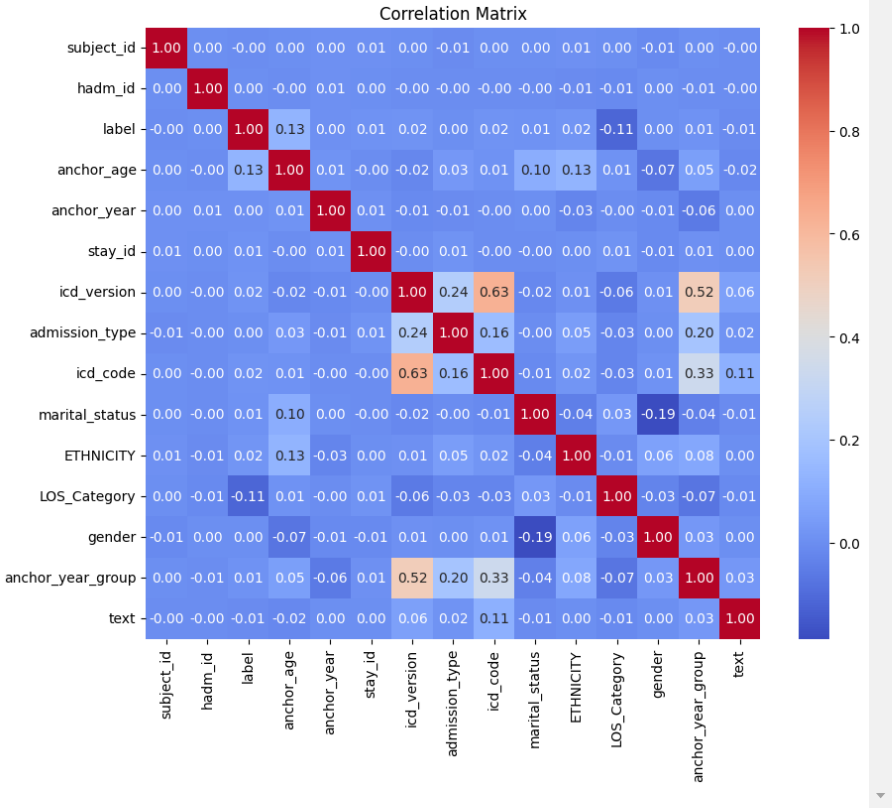









Count
150000
100000
50000
0
2014 - 2016
2011 - 2013
2008 - 2010
2017 - 2019
Patient Year Group


Correlation Matrix
subject_id 1.00 0.00 -0.00 0.00 0.00 0.01 0.00 -0.01 0.00 0.00 0.01 0.00 -0.01 0.00 -0.00
hadm_id 0.00 1.00 0.00 -0.00 0.01 0.00 -0.00 -0.00 -0.00 -0.00 -0.01 -0.01 0.00 -0.01 -0.00
label -0.00 0.00 1.00 0.13 0.00 0.01 0.02 0.00 0.02 0.01 0.02 -0.11 0.00 0.01 -0.01
anchor_age 0.00 -0.00 0.13 1.00 0.01 -0.00 -0.02 0.03 0.01 0.10 0.13 0.01 -0.07 0.05 -0.02
anchor_year 0.00 0.01 0.00 0.01 1.00 0.01 -0.01 -0.01 -0.00 0.00 -0.03 -0.00 -0.01 -0.06 0.00
stay_id 0.01 0.00 0.01 -0.00 0.01 1.00 -0.00 0.01 -0.00 -0.00 0.00 0.01 -0.01 0.01 0.00
icd_version 0.00 -0.00 0.02 -0.02 -0.01 -0.00 1.00 0.24 0.63 -0.02 0.01 -0.06 0.01 0.52 0.06
admission_type -0.01 -0.00 0.00 0.03 -0.01 0.01 0.24 1.00 0.16 -0.00 0.05 -0.03 0.00 0.20 0.02
icd_code 0.00 -0.00 0.02 0.01 -0.00 -0.00 0.63 0.16 1.00 -0.01 0.02 -0.03 0.01 0.33 0.11
marital_status 0.00 -0.00 0.01 0.10 0.00 -0.00 -0.02 -0.00 -0.01 1.00 -0.04 0.03 -0.19 -0.04 -0.01
ETHNICITY 0.01 -0.01 0.02 0.13 -0.03 0.00 0.01 0.05 0.02 -0.04 1.00 -0.01 0.06 0.08 0.00
LOS_Category 0.00 -0.01 -0.11 0.01 -0.00 0.01 -0.06 -0.03 -0.03 0.03 -0.01 1.00 -0.03 -0.07 -0.01
gender -0.01 0.00 0.00 -0.07 -0.01 -0.01 0.01 0.00 0.01 -0.19 0.06 -0.03 1.00 0.03 0.00
anchor_year_group 0.00 -0.01 0.01 0.05 -0.06 0.01 0.52 0.20 0.33 -0.04 0.08 -0.07 0.03 1.00 0.03
text -0.00 -0.00 -0.01 -0.02 0.00 0.00 0.06 0.02 0.11 -0.01 0.00 -0.01 0.00 0.03 1.00
subject_id
hadm_id
label
anchor_age
anchor_year
stay_id
icd_version
admission_type
icd_code
marital_status
ETHNICITY
LOS_Category
gender
anchor_year_group
text
1.0
0.8
0.6
0.4
0.2
0.0

Start coding or generate with AI.

```
from google.colab import drive
import pandas as pd
import numpy as np
import torch
from transformers import BertTokenizer, AdamW, BertForSequenceClassification
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.model_selection import train_test_split
from sklearn.base import BaseEstimator, ClassifierMixin
import optuna
from joblib import dump  # Added import for dump function from joblib
import warnings

# Ignore warnings
warnings.filterwarnings('ignore')

# Function to load data
def load_data():
    # Mount Google Drive
    drive.mount('/content/drive')
    # Load the DataFrames
    folder_path = '/content/drive/My Drive/Hs/'

    # Load the datasets
    admissions = pd.read_csv(folder_path +'admissions.csv')
    patients = pd.read_csv(folder_path +'patients.csv')
    icustays = pd.read_csv(folder_path + 'icustays.csv')
    diagnoses_icd = pd.read_csv(folder_path + 'diagnoses_icd.csv')
    d_icd_diagnoses = pd.read_csv(folder_path + 'd_icd_diagnoses.csv')

    # Merge datasets based on common columns using inner join
    merged_data = pd.merge(admissions, patients, on='subject_id', how='inner')
    merged_data = pd.merge(merged_data, icustays, on=['subject_id', 'hadm_id'], how='inner')
    merged_data = pd.merge(merged_data, diagnoses_icd, on=['subject_id', 'hadm_id'], how='inner')
    merged_data = pd.merge(merged_data, d_icd_diagnoses, on=['icd_code', 'icd_version'], how='inner')
    merged_data.rename(columns={'hospital_expire_flag': 'label', 'long_title': 'text'}, inplace=True)
    merged_data.dropna(inplace=True)
    print(merged_data.isnull().sum())
    merged_data = merged_data.sample(100, random_state=42)
    return merged_data

# Custom BERT Classifier
class MedBERTClassifier(BaseEstimator, ClassifierMixin):
    def __init__(self, model_name='microsoft/BiomedNLP-PubMedBERT-base-uncased-abstract', lr=2e-5, epochs=3, batch_size=32, early_stopping=T
        self.model_name = model_name
        self.lr = lr
        self.epochs = epochs
        self.batch_size = batch_size
        self.early_stopping = early_stopping
        self.patience = patience

    def fit(self, X, y):
        self.model = BertForSequenceClassification.from_pretrained(self.model_name, num_labels=2)
        self.tokenizer = BertTokenizer.from_pretrained(self.model_name)
        self.optimizer = AdamW(self.model.parameters(), lr=self.lr)
        best_val_loss = float('inf')
        patience_counter = 0

        # Tokenize text data and prepare input tensors
        inputs = self.tokenizer(X["text"].tolist(), padding=True, truncation=True, return_tensors='pt', max_length=128)
        labels = torch.tensor(y.tolist())

        # Train the model
        for epoch in range(self.epochs):
            self.model.train()
            running_loss = 0.0
            for i in range(0, len(labels), self.batch_size):
                input_batch = {k: v[i:i+self.batch_size] for k, v in inputs.items()}
                label_batch = labels[i:i+self.batch_size]

                self.optimizer.zero_grad()
                outputs = self.model(**input_batch, labels=label_batch)
                loss = outputs.loss
                loss.backward()
                self.optimizer.step()

                running_loss += loss.item()
```

```
            # Early stopping
            if self.early_stopping:
                val_loss = self.evaluate_loss(X_val, y_val)
                if val_loss < best_val_loss:
                    best_val_loss = val_loss
                    patience_counter = 0
                else:
                    patience_counter += 1
                    if patience_counter >= self.patience:
                        print("Early stopping after {} epochs.".format(epoch + 1))
                        break

    def predict(self, X):
        # Tokenize text data and prepare input tensors
        inputs = self.tokenizer(X["text"].tolist(), padding=True, truncation=True, return_tensors='pt', max_length=128)

        # Evaluate the model
        self.model.eval()
        predictions = []
        with torch.no_grad():
            for i in range(0, len(inputs['input_ids']), self.batch_size):
                input_batch = {k: v[i:i+self.batch_size] for k, v in inputs.items()}
                outputs = self.model(**input_batch)
                logits = outputs.logits
                predictions.extend(logits.argmax(dim=1).cpu().tolist())
        return predictions

    def evaluate_loss(self, X, y):
        inputs = self.tokenizer(X["text"].tolist(), padding=True, truncation=True, return_tensors='pt', max_length=128)
        labels = torch.tensor(y.tolist())

        self.model.eval()
        total_loss = 0.0
        with torch.no_grad():
            for i in range(0, len(labels), self.batch_size):
                input_batch = {k: v[i:i+self.batch_size] for k, v in inputs.items()}
                label_batch = labels[i:i+self.batch_size]
                outputs = self.model(**input_batch, labels=label_batch)
                loss = outputs.loss
                total_loss += loss.item()

        return total_loss / (len(labels) / self.batch_size)

# Define objective function for Optuna
def objective(trial):
    # Define parameters to search
    lr = trial.suggest_loguniform('lr', 1e-6, 1e-4)
    epochs = trial.suggest_int('epochs', 3, 5)
    batch_size = trial.suggest_categorical('batch_size', [16, 32, 64])

    # Initialize MedBERT Classifier with dynamically passed values
    med_bert_classifier = MedBERTClassifier(
        model_name='microsoft/BiomedNLP-PubMedBERT-base-uncased-abstract',
        lr=lr,
        epochs=epochs,
        batch_size=batch_size
    )

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

    # Evaluate using validation set
    predictions = med_bert_classifier.predict(X_val)
    accuracy = accuracy_score(y_val, predictions)
    return accuracy

# Load data
merged_df = load_data()

# Split data into train, validation, and test sets
X = merged_df.drop(columns=["label"])
y = merged_df["label"]
X_train_val, X_test, y_train_val, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_train_val, y_train_val, test_size=0.111, random_state=42)  # 80% train, 10% validation,

# Run hyperparameter optimization
study = optuna.create_study(direction='maximize')
```

```
study.optimize(objective, n_trials=100)

# Print best parameters and performance
print("Best parameters found: ", study.best_params)
print("Best accuracy on validation set: {:.4f}".format(study.best_value))

# Get the best model parameters
best_params = study.best_params

# Train the best model on the entire training and validation data
best_model = MedBERTClassifier(
    model_name='microsoft/BiomedNLP-PubMedBERT-base-uncased-abstract',
    lr=best_params['lr'],
    epochs=best_params['epochs'],
    batch_size=best_params['batch_size']
)
best_model.fit(X_train_val, y_train_val)

# Save the best model to a pkl file
model_filename = "BiomedNLP-PubMedBERT-base-uncased_Model.pkl"  # You can choose a different filename
dump(best_model, model_filename)
print(f"Best model saved to {model_filename}")

# Evaluate the best model on the test set
test_predictions = best_model.predict(X_test)
test_accuracy = accuracy_score(y_test, test_predictions)
test_precision = precision_score(y_test, test_predictions)
test_recall = recall_score(y_test, test_predictions)
test_f1 = f1_score(y_test, test_predictions)

print("Accuracy on test set: {:.4f}".format(test_accuracy))
print("Precision on test set: {:.4f}".format(test_precision))
print("Recall on test set: {:.4f}".format(test_recall))
print("F1-score on test set: {:.4f}".format(test_f1))
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
[I 2024-04-17 16:46:15,713] A new study created in memory with name: no-name-7a98fa35-3e
subject_id            0
hadm_id               0
admittime             0
dischtime             0
deathtime             0
admission_type        0
admit_provider_id     0
admission_location    0
discharge_location    0
insurance             0
language              0
marital_status        0
ETHNICITY             0
edregtime             0
edouttime             0
label                 0
gender                0
anchor_age            0
anchor_year           0
anchor_year_group     0
dod                   0
stay_id               0
first_careunit        0
last_careunit         0
intime                0
outtime               0
los                   0
seq_num               0
icd_code              0
icd_version           0
text                  0
dtype: int64
config.json: 100%                                   385/385 [00:00<00:00, 39.6kB/s]
pytorch_model.bin: 100%                             440M/440M [00:05<00:00, 91.4MB/s]
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
tokenizer_config.json: 100%                         28.0/28.0 [00:00<00:00, 2.68kB/s]
vocab.txt: 100%                                     225k/225k [00:00<00:00, 4.90MB/s]
[I 2024-04-17 16:46:30,330] Trial 0 finished with value: 1.0 and parameters: {'lr': 4.57
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:46:36,446] Trial 1 finished with value: 1.0 and parameters: {'lr': 1.23
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:46:40,457] Trial 2 finished with value: 1.0 and parameters: {'lr': 9.49
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:46:45,420] Trial 3 finished with value: 1.0 and parameters: {'lr': 2.64
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:46:49,587] Trial 4 finished with value: 1.0 and parameters: {'lr': 2.81
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:46:56,996] Trial 5 finished with value: 1.0 and parameters: {'lr': 1.64
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:47:01,163] Trial 6 finished with value: 1.0 and parameters: {'lr': 6.92
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:47:06,297] Trial 7 finished with value: 1.0 and parameters: {'lr': 3.71
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:47:15,704] Trial 8 finished with value: 1.0 and parameters: {'lr': 6.67
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:47:20,974] Trial 9 finished with value: 1.0 and parameters: {'lr': 1.16
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:47:28,253] Trial 10 finished with value: 1.0 and parameters: {'lr': 4.3
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:47:34,251] Trial 11 finished with value: 1.0 and parameters: {'lr': 1.7
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:47:41,695] Trial 12 finished with value: 1.0 and parameters: {'lr': 6.8
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:47:47,673] Trial 13 finished with value: 1.0 and parameters: {'lr': 3.5
Some weights of BertForSequenceClassification were not initialized from the model checkp
```

```
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:47:57,333] Trial 14 finished with value: 1.0 and parameters: {'lr': 9.8
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:48:03,296] Trial 15 finished with value: 1.0 and parameters: {'lr': 1.9
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:48:09,435] Trial 16 finished with value: 1.0 and parameters: {'lr': 5.1
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:48:16,763] Trial 17 finished with value: 1.0 and parameters: {'lr': 4.7
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:48:22,823] Trial 18 finished with value: 1.0 and parameters: {'lr': 1.1
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:48:28,943] Trial 19 finished with value: 1.0 and parameters: {'lr': 1.3
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:48:36,272] Trial 20 finished with value: 1.0 and parameters: {'lr': 2.2
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:48:40,902] Trial 21 finished with value: 1.0 and parameters: {'lr': 9.4
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:48:45,537] Trial 22 finished with value: 1.0 and parameters: {'lr': 9.1
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:48:51,827] Trial 23 finished with value: 1.0 and parameters: {'lr': 2.4
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:48:56,701] Trial 24 finished with value: 1.0 and parameters: {'lr': 6.7
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:49:01,701] Trial 25 finished with value: 1.0 and parameters: {'lr': 4.2
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:49:08,869] Trial 26 finished with value: 1.0 and parameters: {'lr': 2.9
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:49:13,957] Trial 27 finished with value: 1.0 and parameters: {'lr': 1.6
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:49:21,883] Trial 28 finished with value: 1.0 and parameters: {'lr': 4.5
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:49:27,183] Trial 29 finished with value: 1.0 and parameters: {'lr': 2.3
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:49:32,138] Trial 30 finished with value: 0.9 and parameters: {'lr': 7.1
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:49:37,175] Trial 31 finished with value: 1.0 and parameters: {'lr': 1.2
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:49:41,567] Trial 32 finished with value: 1.0 and parameters: {'lr': 2.4
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:49:46,562] Trial 33 finished with value: 1.0 and parameters: {'lr': 5.4
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:49:50,739] Trial 34 finished with value: 1.0 and parameters: {'lr': 2.8
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:49:56,976] Trial 35 finished with value: 1.0 and parameters: {'lr': 8.1
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:50:02,943] Trial 36 finished with value: 1.0 and parameters: {'lr': 3.8
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:50:09,007] Trial 37 finished with value: 1.0 and parameters: {'lr': 2.5
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:50:13,184] Trial 38 finished with value: 1.0 and parameters: {'lr': 5.8
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:50:20,312] Trial 39 finished with value: 1.0 and parameters: {'lr': 1.0
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:50:28,399] Trial 40 finished with value: 1.0 and parameters: {'lr': 1.6
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:50:32,668] Trial 41 finished with value: 1.0 and parameters: {'lr': 2.2
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
```

You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:50:37,029] Trial 42 finished with value: 0.5 and parameters: {'lr': 3.1
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:50:41,218] Trial 43 finished with value: 1.0 and parameters: {'lr': 1.1
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:50:47,222] Trial 44 finished with value: 0.0 and parameters: {'lr': 1.4
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:50:52,265] Trial 45 finished with value: 1.0 and parameters: {'lr': 5.5
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:50:57,313] Trial 46 finished with value: 0.9 and parameters: {'lr': 1.7
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:51:03,535] Trial 47 finished with value: 1.0 and parameters: {'lr': 3.9
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:51:08,187] Trial 48 finished with value: 1.0 and parameters: {'lr': 7.8
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:51:15,904] Trial 49 finished with value: 1.0 and parameters: {'lr': 1.9
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:51:21,822] Trial 50 finished with value: 1.0 and parameters: {'lr': 8.3
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:51:28,996] Trial 51 finished with value: 1.0 and parameters: {'lr': 1.5
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:51:36,049] Trial 52 finished with value: 1.0 and parameters: {'lr': 3.4
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:51:43,296] Trial 53 finished with value: 0.7 and parameters: {'lr': 2.0
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:51:50,455] Trial 54 finished with value: 1.0 and parameters: {'lr': 3.1
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:51:57,651] Trial 55 finished with value: 1.0 and parameters: {'lr': 3.7
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:52:03,880] Trial 56 finished with value: 1.0 and parameters: {'lr': 5.3
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:52:11,187] Trial 57 finished with value: 1.0 and parameters: {'lr': 1.3
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:52:18,955] Trial 58 finished with value: 0.0 and parameters: {'lr': 1.2
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:52:24,865] Trial 59 finished with value: 1.0 and parameters: {'lr': 2.8
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:52:29,566] Trial 60 finished with value: 1.0 and parameters: {'lr': 1.7
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:52:33,657] Trial 61 finished with value: 1.0 and parameters: {'lr': 6.7
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:52:37,890] Trial 62 finished with value: 1.0 and parameters: {'lr': 9.1
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:52:41,884] Trial 63 finished with value: 1.0 and parameters: {'lr': 4.3
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:52:45,856] Trial 64 finished with value: 1.0 and parameters: {'lr': 6.5
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:52:50,131] Trial 65 finished with value: 1.0 and parameters: {'lr': 5.2
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:52:54,287] Trial 66 finished with value: 1.0 and parameters: {'lr': 1.8
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:53:02,452] Trial 67 finished with value: 1.0 and parameters: {'lr': 3.5
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:53:06,551] Trial 68 finished with value: 0.2 and parameters: {'lr': 1.0
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:53:14,187] Trial 69 finished with value: 1.0 and parameters: {'lr': 1.0
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:53:19,264] Trial 70 finished with value: 1.0 and parameters: {'lr': 8.4

[I 2024-04-17 16:53:19,284] Trial 70 finished with value: 1.0 and parameters: {'lr': 8.2
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:53:24,387] Trial 71 finished with value: 1.0 and parameters: {'lr': 2.2
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:53:29,383] Trial 72 finished with value: 1.0 and parameters: {'lr': 2.6
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:53:34,672] Trial 73 finished with value: 1.0 and parameters: {'lr': 3.1
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:53:39,610] Trial 74 finished with value: 1.0 and parameters: {'lr': 4.2
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:53:44,605] Trial 75 finished with value: 1.0 and parameters: {'lr': 4.7
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:53:49,987] Trial 76 finished with value: 1.0 and parameters: {'lr': 4.4
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:53:56,014] Trial 77 finished with value: 1.0 and parameters: {'lr': 1.4
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:54:01,098] Trial 78 finished with value: 1.0 and parameters: {'lr': 6.4
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:54:07,167] Trial 79 finished with value: 1.0 and parameters: {'lr': 1.1
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:54:13,197] Trial 80 finished with value: 1.0 and parameters: {'lr': 6.7
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:54:22,946] Trial 81 finished with value: 1.0 and parameters: {'lr': 7.7
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:54:32,246] Trial 82 finished with value: 1.0 and parameters: {'lr': 9.8
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:54:42,384] Trial 83 finished with value: 1.0 and parameters: {'lr': 5.4
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:54:52,323] Trial 84 finished with value: 1.0 and parameters: {'lr': 3.4
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:55:02,140] Trial 85 finished with value: 1.0 and parameters: {'lr': 5.7
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:55:08,902] Trial 86 finished with value: 1.0 and parameters: {'lr': 4.9
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:55:12,881] Trial 87 finished with value: 1.0 and parameters: {'lr': 7.3
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:55:20,298] Trial 88 finished with value: 1.0 and parameters: {'lr': 8.6
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:55:26,974] Trial 89 finished with value: 1.0 and parameters: {'lr': 6.0
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:55:33,241] Trial 90 finished with value: 1.0 and parameters: {'lr': 2.7
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:55:37,572] Trial 91 finished with value: 1.0 and parameters: {'lr': 9.6
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:55:43,217] Trial 92 finished with value: 1.0 and parameters: {'lr': 7.6
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:55:47,312] Trial 93 finished with value: 1.0 and parameters: {'lr': 2.1
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:55:53,840] Trial 94 finished with value: 1.0 and parameters: {'lr': 3.9
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:55:59,095] Trial 95 finished with value: 1.0 and parameters: {'lr': 1.6
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:56:05,385] Trial 96 finished with value: 1.0 and parameters: {'lr': 1.1
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:56:12,731] Trial 97 finished with value: 1.0 and parameters: {'lr': 7.0
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:56:16,924] Trial 98 finished with value: 1.0 and parameters: {'lr': 5.9

```
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
[I 2024-04-17 16:56:22,126] Trial 99 finished with value: 0.9 and parameters: {'lr': 1.9
Best parameters found:  {'lr': 4.5745364075126895e-05, 'epochs': 5, 'batch_size': 32}
Best accuracy on validation set: 1.0000
Some weights of BertForSequenceClassification were not initialized from the model checkp
You should probably TRAIN this model on a down-stream task to be able to use it for prec
Best model saved to BiomedNLP-PubMedBERT-base-uncased_Model.pkl
Accuracy on test set: 1.0000
Precision on test set: 1.0000
Recall on test set: 1.0000
F1-score on test set: 1.0000
```

```
from google.colab import drive
import pandas as pd
import numpy as np
import torch
from transformers import BertTokenizer, AdamW, BertForSequenceClassification
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.model_selection import train_test_split
from sklearn.base import BaseEstimator, ClassifierMixin
import optuna
from joblib import dump  # Added import for dump function from joblib
import warnings

# Ignore warnings
warnings.filterwarnings('ignore')

# Function to load data
def load_data():
    # Mount Google Drive
    drive.mount('/content/drive')
    # Load the DataFrames
    folder_path = '/content/drive/My Drive/Hs/'

    # Load the datasets
    admissions = pd.read_csv(folder_path +'admissions.csv')
    patients = pd.read_csv(folder_path +'patients.csv')
    icustays = pd.read_csv(folder_path + 'icustays.csv')
    diagnoses_icd = pd.read_csv(folder_path + 'diagnoses_icd.csv')
    d_icd_diagnoses = pd.read_csv(folder_path + 'd_icd_diagnoses.csv')

    # Merge datasets based on common columns using inner join
    merged_data = pd.merge(admissions, patients, on='subject_id', how='inner')
    merged_data = pd.merge(merged_data, icustays, on=['subject_id', 'hadm_id'], how='inner')
    merged_data = pd.merge(merged_data, diagnoses_icd, on=['subject_id', 'hadm_id'], how='inner')
    merged_data = pd.merge(merged_data, d_icd_diagnoses, on=['icd_code', 'icd_version'], how='inner')
    merged_data.rename(columns={'hospital_expire_flag': 'label', 'long_title': 'text'}, inplace=True)
    merged_data.dropna(inplace=True)
    print(merged_data.isnull().sum())
    merged_data = merged_data.sample(100, random_state=42)
    return merged_data

# Custom BERT Classifier
class BioBERTClassifier(BaseEstimator, ClassifierMixin):
    def __init__(self, model_name='dmis-lab/biobert-v1.1', lr=2e-5, epochs=3, batch_size=32, early_stopping=True, patience=3):
        self.model_name = model_name
        self.lr = lr
        self.epochs = epochs
        self.batch_size = batch_size
        self.early_stopping = early_stopping
        self.patience = patience

    def fit(self, X, y):
        self.model = BertForSequenceClassification.from_pretrained(self.model_name, num_labels=2)
        self.tokenizer = BertTokenizer.from_pretrained(self.model_name)
        self.optimizer = AdamW(self.model.parameters(), lr=self.lr)
        best_val_loss = float('inf')
        patience_counter = 0

        # Tokenize text data and prepare input tensors
        inputs = self.tokenizer(X["text"].tolist(), padding=True, truncation=True, return_tensors='pt', max_length=128)
        labels = torch.tensor(y.tolist())

        # Train the model
        for epoch in range(self.epochs):
```