Lab Logbook Requirement:

1. *Create and train your own LSTM model*
2. *Add all the LSTM's Error metrics: Accuracy, Precision, Recall, F1-Score and AUC to the final histogram "ML Models performance...".*

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
import matplotlib.pyplot as plt


credit_data = pd.read_csv("credit_risk_dataset.csv")


credit_data['person_home_ownership'] = LabelEncoder().fit_transform(credit_data['person_home_ownership'])
credit_data['loan_intent'] = LabelEncoder().fit_transform(credit_data['loan_intent'])
credit_data['cb_person_default_on_file'] = LabelEncoder().fit_transform(credit_data['cb_person_default_on_file'])


credit_data.fillna(credit_data.median(), inplace=True)


X = credit_data.drop('loan_status', axis=1).values
y = credit_data['loan_status'].values


scaler = StandardScaler()
X = scaler.fit_transform(X)


X = X.reshape((X.shape[0], 1, X.shape[1]))
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


model = Sequential([
    LSTM(50, input_shape=(X_train.shape[1], X_train.shape[2]), activation='relu'),
    Dense(1, activation='sigmoid')
])


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


history = model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=1, validation_data=(X_test, y_test))


y_pred = (model.predict(X_test) > 0.5).astype(int)


accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc = roc_auc_score(y_test, model.predict(X_test))
```

```
metrics = {
    'Accuracy': accuracy,
    'Precision': precision,
    'Recall': recall,
    'F1-Score': f1,
    'AUC': auc
}


plt.bar(metrics.keys(), metrics.values())
plt.title("ML Models Performance")
plt.ylabel("Score")
plt.show()
```

```
Epoch 1/10
815/815 ━━━━━━━━━━━━━━━━━━━━ 18s 7ms/step - accuracy: 0.7968 - loss: 0.4926 - val_accuracy: 0.8373 - val_loss: 0.3728
Epoch 2/10
815/815 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - accuracy: 0.8531 - loss: 0.3549 - val_accuracy: 0.8558 - val_loss: 0.3509
Epoch 3/10
815/815 ━━━━━━━━━━━━━━━━━━━━ 4s 5ms/step - accuracy: 0.8613 - loss: 0.3392 - val_accuracy: 0.8599 - val_loss: 0.3430
Epoch 4/10
815/815 ━━━━━━━━━━━━━━━━━━━━ 4s 5ms/step - accuracy: 0.8677 - loss: 0.3317 - val_accuracy: 0.8601 - val_loss: 0.3385
Epoch 5/10
815/815 ━━━━━━━━━━━━━━━━━━━━ 4s 5ms/step - accuracy: 0.8646 - loss: 0.3358 - val_accuracy: 0.8665 - val_loss: 0.3320
Epoch 6/10
815/815 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - accuracy: 0.8702 - loss: 0.3224 - val_accuracy: 0.8690 - val_loss: 0.3287
Epoch 7/10
815/815 ━━━━━━━━━━━━━━━━━━━━ 5s 6ms/step - accuracy: 0.8682 - loss: 0.3284 - val_accuracy: 0.8668 - val_loss: 0.3277
Epoch 8/10
815/815 ━━━━━━━━━━━━━━━━━━━━ 4s 5ms/step - accuracy: 0.8719 - loss: 0.3188 - val_accuracy: 0.8700 - val_loss: 0.3238
Epoch 9/10
815/815 ━━━━━━━━━━━━━━━━━━━━ 5s 6ms/step - accuracy: 0.8742 - loss: 0.3157 - val_accuracy: 0.8682 - val_loss: 0.3220
Epoch 10/10
815/815 ━━━━━━━━━━━━━━━━━━━━ 5s 5ms/step - accuracy: 0.8783 - loss: 0.3084 - val_accuracy: 0.8752 - val_loss: 0.3207
204/204 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step
204/204 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
```