Chloy_116_CIA_1PG_1

October 8, 2025

[8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[9]:
```
data = pd.read_csv('CIA1_Dataset.csv')

data.head()
```

[9]:

	income	loan_amount	credit_score	age	employment_years	approved
0	57450.71	206927.18	5.09	57	22	0
1	47926.04	187545.82	5.64	40	39	0
2	59715.33	293422.56	5.40	41	10	0
3	72845.45	220353.67	7.00	30	31	0
4	46487.70	227008.21	2.26	25	35	0

[10]:
```
data.isna().sum()
```

[10]:
```
income              0
loan_amount         0
credit_score        0
age                 0
employment_years    0
approved            0
dtype: int64
```

0.1 Single Layer Perceptron to classify Loan Approvals

[11]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Perceptron
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix

X = data[['credit_score', 'age', 'employment_years']]
y = data['approved']
```

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

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

perceptron = Perceptron(random_state=42, max_iter=1000)
perceptron.fit(X_train_scaled, y_train)

y_pred = perceptron.predict(X_test_scaled)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.4f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.9545

Classification Report:
              precision    recall  f1-score   support

           0       0.95      1.00      0.98       105
           1       0.00      0.00      0.00         5

    accuracy                           0.95       110
   macro avg       0.48      0.50      0.49       110
weighted avg       0.91      0.95      0.93       110


Confusion Matrix:
[[105   0]
 [  5   0]]

/home/chloycosta/Documents/College_code/Sem_5/NNDL/.venv/lib64/python3.11/site-
packages/sklearn/metrics/_classification.py:1731: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
/home/chloycosta/Documents/College_code/Sem_5/NNDL/.venv/lib64/python3.11/site-
packages/sklearn/metrics/_classification.py:1731: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
/home/chloycosta/Documents/College_code/Sem_5/NNDL/.venv/lib64/python3.11/site-
packages/sklearn/metrics/_classification.py:1731: UndefinedMetricWarning:
```

```
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
```

0.2 Feed forward Neural Network with one hidden layer

[12]:
```
from tensorflow import keras
from tensorflow.keras import layers

keras_model = keras.Sequential([
    layers.Dense(10, activation='relu', input_shape=(3,)),
    layers.Dense(1, activation='sigmoid')
])

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

history = keras_model.fit(
    X_train_scaled, y_train,
    epochs=100,
    batch_size=32,
    validation_split=0.2,
    verbose=1
)

y_pred_keras = (keras_model.predict(X_test_scaled) > 0.5).astype(int).flatten()

accuracy_keras = accuracy_score(y_test, y_pred_keras)
print(f"Keras Neural Network Accuracy: {accuracy_keras:.4f}")
print("\nKeras Classification Report:")
print(classification_report(y_test, y_pred_keras))
print("\nKeras Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_keras))
```

```
Epoch 1/100

/home/chloycosta/Documents/College_code/Sem_5/NNDL/.venv/lib64/python3.11/site-
packages/keras/src/layers/core/dense.py:92: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

11/11             1s 16ms/step -
accuracy: 0.2017 - loss: 0.9839 - val_accuracy: 0.2273 - val_loss: 0.9182
Epoch 2/100
11/11             0s 6ms/step -
```

```
accuracy: 0.2358 - loss: 0.9418 - val_accuracy: 0.2386 - val_loss: 0.8819
Epoch 3/100
11/11             0s 7ms/step -
accuracy: 0.2614 - loss: 0.9037 - val_accuracy: 0.2955 - val_loss: 0.8475
Epoch 4/100
11/11             0s 6ms/step -
accuracy: 0.2869 - loss: 0.8677 - val_accuracy: 0.3750 - val_loss: 0.8148
Epoch 5/100
11/11             0s 8ms/step -
accuracy: 0.3097 - loss: 0.8332 - val_accuracy: 0.4091 - val_loss: 0.7851
Epoch 6/100
11/11             0s 6ms/step -
accuracy: 0.3494 - loss: 0.8023 - val_accuracy: 0.4545 - val_loss: 0.7565
Epoch 7/100
11/11             0s 5ms/step -
accuracy: 0.3835 - loss: 0.7719 - val_accuracy: 0.5114 - val_loss: 0.7306
Epoch 8/100
11/11             0s 6ms/step -
accuracy: 0.4375 - loss: 0.7449 - val_accuracy: 0.5795 - val_loss: 0.7060
Epoch 9/100
11/11             0s 6ms/step -
accuracy: 0.4915 - loss: 0.7180 - val_accuracy: 0.6023 - val_loss: 0.6839
Epoch 10/100
11/11             0s 7ms/step -
accuracy: 0.5369 - loss: 0.6947 - val_accuracy: 0.6477 - val_loss: 0.6623
Epoch 11/100
11/11             0s 6ms/step -
accuracy: 0.5881 - loss: 0.6713 - val_accuracy: 0.6818 - val_loss: 0.6426
Epoch 12/100
11/11             0s 6ms/step -
accuracy: 0.6278 - loss: 0.6507 - val_accuracy: 0.7045 - val_loss: 0.6235
Epoch 13/100
11/11             0s 6ms/step -
accuracy: 0.6477 - loss: 0.6306 - val_accuracy: 0.7273 - val_loss: 0.6059
Epoch 14/100
11/11             0s 6ms/step -
accuracy: 0.6562 - loss: 0.6114 - val_accuracy: 0.7500 - val_loss: 0.5898
Epoch 15/100
11/11             0s 7ms/step -
accuracy: 0.6960 - loss: 0.5943 - val_accuracy: 0.7500 - val_loss: 0.5739
Epoch 16/100
11/11             0s 6ms/step -
accuracy: 0.7159 - loss: 0.5776 - val_accuracy: 0.7614 - val_loss: 0.5591
Epoch 17/100
11/11             0s 6ms/step -
accuracy: 0.7415 - loss: 0.5618 - val_accuracy: 0.7727 - val_loss: 0.5451
Epoch 18/100
11/11             0s 6ms/step -
```

```
accuracy: 0.7585 - loss: 0.5466 - val_accuracy: 0.8182 - val_loss: 0.5316
Epoch 19/100
11/11             0s 6ms/step -
accuracy: 0.7756 - loss: 0.5322 - val_accuracy: 0.8182 - val_loss: 0.5187
Epoch 20/100
11/11             0s 5ms/step -
accuracy: 0.7955 - loss: 0.5182 - val_accuracy: 0.8409 - val_loss: 0.5064
Epoch 21/100
11/11             0s 6ms/step -
accuracy: 0.8352 - loss: 0.5050 - val_accuracy: 0.8523 - val_loss: 0.4944
Epoch 22/100
11/11             0s 6ms/step -
accuracy: 0.8580 - loss: 0.4920 - val_accuracy: 0.8750 - val_loss: 0.4829
Epoch 23/100
11/11             0s 6ms/step -
accuracy: 0.8722 - loss: 0.4799 - val_accuracy: 0.8750 - val_loss: 0.4716
Epoch 24/100
11/11             0s 6ms/step -
accuracy: 0.8807 - loss: 0.4679 - val_accuracy: 0.8864 - val_loss: 0.4608
Epoch 25/100
11/11             0s 6ms/step -
accuracy: 0.8977 - loss: 0.4563 - val_accuracy: 0.8864 - val_loss: 0.4500
Epoch 26/100
11/11             0s 6ms/step -
accuracy: 0.9062 - loss: 0.4450 - val_accuracy: 0.8977 - val_loss: 0.4394
Epoch 27/100
11/11             0s 7ms/step -
accuracy: 0.9176 - loss: 0.4341 - val_accuracy: 0.9091 - val_loss: 0.4287
Epoch 28/100
11/11             0s 6ms/step -
accuracy: 0.9261 - loss: 0.4233 - val_accuracy: 0.9318 - val_loss: 0.4184
Epoch 29/100
11/11             0s 6ms/step -
accuracy: 0.9347 - loss: 0.4125 - val_accuracy: 0.9318 - val_loss: 0.4084
Epoch 30/100
11/11             0s 6ms/step -
accuracy: 0.9432 - loss: 0.4021 - val_accuracy: 0.9432 - val_loss: 0.3985
Epoch 31/100
11/11             0s 6ms/step -
accuracy: 0.9489 - loss: 0.3918 - val_accuracy: 0.9432 - val_loss: 0.3890
Epoch 32/100
11/11             0s 6ms/step -
accuracy: 0.9517 - loss: 0.3820 - val_accuracy: 0.9545 - val_loss: 0.3795
Epoch 33/100
11/11             0s 6ms/step -
accuracy: 0.9517 - loss: 0.3723 - val_accuracy: 0.9545 - val_loss: 0.3702
Epoch 34/100
11/11             0s 6ms/step -
```

```
accuracy: 0.9545 - loss: 0.3629 - val_accuracy: 0.9545 - val_loss: 0.3611
Epoch 35/100
11/11             0s 7ms/step -
accuracy: 0.9545 - loss: 0.3539 - val_accuracy: 0.9545 - val_loss: 0.3520
Epoch 36/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.3448 - val_accuracy: 0.9545 - val_loss: 0.3432
Epoch 37/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.3365 - val_accuracy: 0.9545 - val_loss: 0.3343
Epoch 38/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.3277 - val_accuracy: 0.9545 - val_loss: 0.3259
Epoch 39/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.3195 - val_accuracy: 0.9545 - val_loss: 0.3175
Epoch 40/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.3114 - val_accuracy: 0.9545 - val_loss: 0.3096
Epoch 41/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.3037 - val_accuracy: 0.9545 - val_loss: 0.3017
Epoch 42/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.2961 - val_accuracy: 0.9545 - val_loss: 0.2943
Epoch 43/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.2891 - val_accuracy: 0.9545 - val_loss: 0.2867
Epoch 44/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.2819 - val_accuracy: 0.9545 - val_loss: 0.2796
Epoch 45/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.2752 - val_accuracy: 0.9545 - val_loss: 0.2727
Epoch 46/100
11/11             0s 5ms/step -
accuracy: 0.9545 - loss: 0.2687 - val_accuracy: 0.9545 - val_loss: 0.2660
Epoch 47/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.2623 - val_accuracy: 0.9545 - val_loss: 0.2598
Epoch 48/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.2563 - val_accuracy: 0.9545 - val_loss: 0.2536
Epoch 49/100
11/11             0s 5ms/step -
accuracy: 0.9545 - loss: 0.2505 - val_accuracy: 0.9545 - val_loss: 0.2477
Epoch 50/100
11/11             0s 6ms/step -
```

```
accuracy: 0.9545 - loss: 0.2449 - val_accuracy: 0.9545 - val_loss: 0.2420
Epoch 51/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.2395 - val_accuracy: 0.9545 - val_loss: 0.2365
Epoch 52/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.2342 - val_accuracy: 0.9545 - val_loss: 0.2313
Epoch 53/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.2296 - val_accuracy: 0.9545 - val_loss: 0.2261
Epoch 54/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.2247 - val_accuracy: 0.9545 - val_loss: 0.2213
Epoch 55/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.2201 - val_accuracy: 0.9545 - val_loss: 0.2167
Epoch 56/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.2160 - val_accuracy: 0.9545 - val_loss: 0.2120
Epoch 57/100
11/11             0s 9ms/step -
accuracy: 0.9545 - loss: 0.2117 - val_accuracy: 0.9545 - val_loss: 0.2077
Epoch 58/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.2077 - val_accuracy: 0.9545 - val_loss: 0.2037
Epoch 59/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.2039 - val_accuracy: 0.9545 - val_loss: 0.1999
Epoch 60/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.2005 - val_accuracy: 0.9545 - val_loss: 0.1960
Epoch 61/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1971 - val_accuracy: 0.9545 - val_loss: 0.1925
Epoch 62/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1938 - val_accuracy: 0.9545 - val_loss: 0.1891
Epoch 63/100
11/11             0s 5ms/step -
accuracy: 0.9545 - loss: 0.1908 - val_accuracy: 0.9545 - val_loss: 0.1860
Epoch 64/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1879 - val_accuracy: 0.9545 - val_loss: 0.1830
Epoch 65/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1851 - val_accuracy: 0.9545 - val_loss: 0.1801
Epoch 66/100
11/11             0s 7ms/step -
```

```
accuracy: 0.9545 - loss: 0.1824 - val_accuracy: 0.9545 - val_loss: 0.1773
Epoch 67/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1800 - val_accuracy: 0.9545 - val_loss: 0.1746
Epoch 68/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1775 - val_accuracy: 0.9545 - val_loss: 0.1722
Epoch 69/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1752 - val_accuracy: 0.9545 - val_loss: 0.1698
Epoch 70/100
11/11             0s 7ms/step -
accuracy: 0.9545 - loss: 0.1731 - val_accuracy: 0.9545 - val_loss: 0.1675
Epoch 71/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1711 - val_accuracy: 0.9545 - val_loss: 0.1652
Epoch 72/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1691 - val_accuracy: 0.9545 - val_loss: 0.1629
Epoch 73/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1673 - val_accuracy: 0.9545 - val_loss: 0.1608
Epoch 74/100
11/11             0s 7ms/step -
accuracy: 0.9545 - loss: 0.1655 - val_accuracy: 0.9545 - val_loss: 0.1589
Epoch 75/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1638 - val_accuracy: 0.9545 - val_loss: 0.1571
Epoch 76/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1622 - val_accuracy: 0.9545 - val_loss: 0.1553
Epoch 77/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1607 - val_accuracy: 0.9545 - val_loss: 0.1535
Epoch 78/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1593 - val_accuracy: 0.9545 - val_loss: 0.1518
Epoch 79/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1580 - val_accuracy: 0.9545 - val_loss: 0.1501
Epoch 80/100
11/11             0s 7ms/step -
accuracy: 0.9545 - loss: 0.1566 - val_accuracy: 0.9545 - val_loss: 0.1486
Epoch 81/100
11/11             0s 7ms/step -
accuracy: 0.9545 - loss: 0.1553 - val_accuracy: 0.9545 - val_loss: 0.1470
Epoch 82/100
11/11             0s 6ms/step -
```

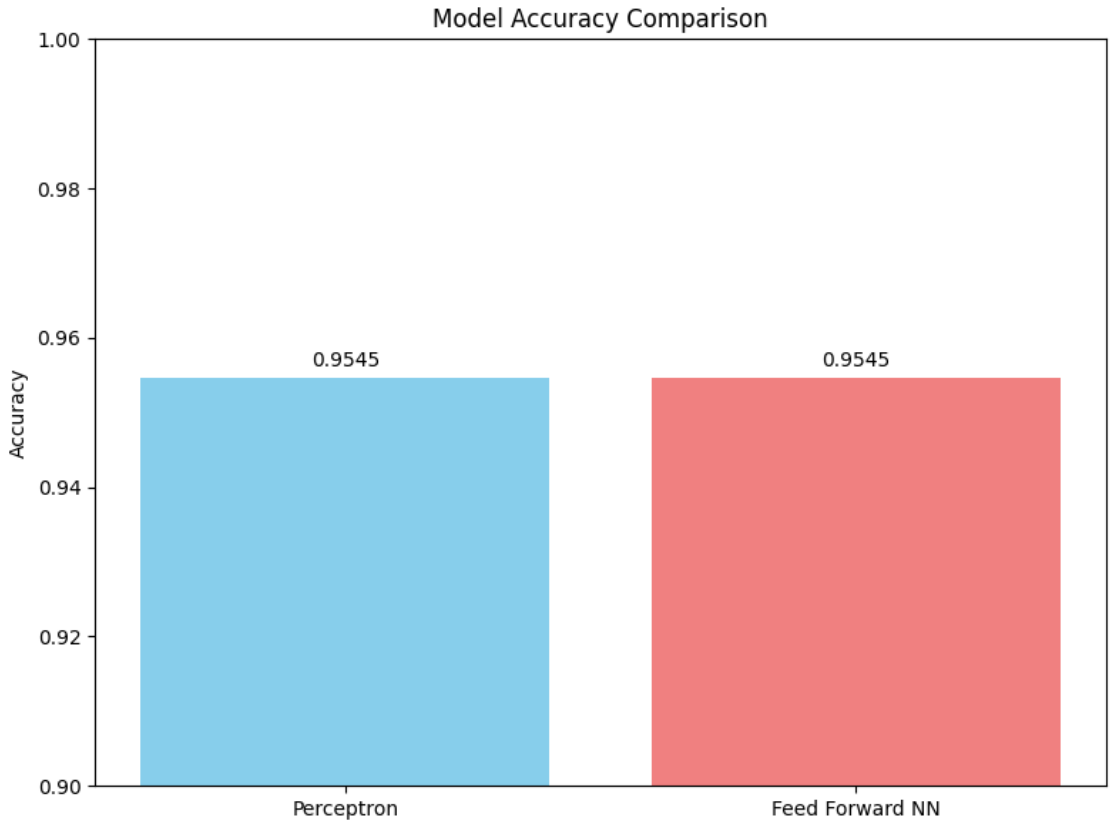
```
accuracy: 0.9545 - loss: 0.1541 - val_accuracy: 0.9545 - val_loss: 0.1456
Epoch 83/100
11/11             0s 7ms/step -
accuracy: 0.9545 - loss: 0.1529 - val_accuracy: 0.9545 - val_loss: 0.1441
Epoch 84/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1519 - val_accuracy: 0.9545 - val_loss: 0.1428
Epoch 85/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1508 - val_accuracy: 0.9545 - val_loss: 0.1415
Epoch 86/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1497 - val_accuracy: 0.9545 - val_loss: 0.1403
Epoch 87/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1488 - val_accuracy: 0.9545 - val_loss: 0.1390
Epoch 88/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1479 - val_accuracy: 0.9545 - val_loss: 0.1378
Epoch 89/100
11/11             0s 5ms/step -
accuracy: 0.9545 - loss: 0.1469 - val_accuracy: 0.9545 - val_loss: 0.1366
Epoch 90/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1461 - val_accuracy: 0.9545 - val_loss: 0.1355
Epoch 91/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1452 - val_accuracy: 0.9545 - val_loss: 0.1344
Epoch 92/100
11/11             0s 7ms/step -
accuracy: 0.9545 - loss: 0.1443 - val_accuracy: 0.9545 - val_loss: 0.1334
Epoch 93/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1435 - val_accuracy: 0.9545 - val_loss: 0.1324
Epoch 94/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1429 - val_accuracy: 0.9545 - val_loss: 0.1315
Epoch 95/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1421 - val_accuracy: 0.9545 - val_loss: 0.1305
Epoch 96/100
11/11             0s 6ms/step -
accuracy: 0.9545 - loss: 0.1414 - val_accuracy: 0.9545 - val_loss: 0.1296
Epoch 97/100
11/11             0s 7ms/step -
accuracy: 0.9545 - loss: 0.1407 - val_accuracy: 0.9545 - val_loss: 0.1287
Epoch 98/100
11/11             0s 6ms/step -
```

```
accuracy: 0.9545 - loss: 0.1401 - val_accuracy: 0.9545 - val_loss: 0.1278
Epoch 99/100
11/11            0s 6ms/step -
accuracy: 0.9545 - loss: 0.1394 - val_accuracy: 0.9545 - val_loss: 0.1270
Epoch 100/100
11/11            0s 5ms/step -
accuracy: 0.9545 - loss: 0.1388 - val_accuracy: 0.9545 - val_loss: 0.1261
4/4            0s 11ms/step
Keras Neural Network Accuracy: 0.9545

Keras Classification Report:
              precision    recall  f1-score   support

           0       0.95      1.00      0.98       105
           1       0.00      0.00      0.00         5

    accuracy                           0.95       110
   macro avg       0.48      0.50      0.49       110
weighted avg       0.91      0.95      0.93       110


Keras Confusion Matrix:
[[105   0]
 [  5   0]]

/home/chloycosta/Documents/College_code/Sem_5/NNDL/.venv/lib64/python3.11/site-
packages/sklearn/metrics/_classification.py:1731: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
/home/chloycosta/Documents/College_code/Sem_5/NNDL/.venv/lib64/python3.11/site-
packages/sklearn/metrics/_classification.py:1731: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
/home/chloycosta/Documents/College_code/Sem_5/NNDL/.venv/lib64/python3.11/site-
packages/sklearn/metrics/_classification.py:1731: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
```

[13]:
```python
print("Model Accuracy Comparison:")
print(f"Perceptron Accuracy: {accuracy:.4f}")
print(f"Keras Neural Network Accuracy: {accuracy_keras:.4f}")
print(f"Difference: {abs(accuracy - accuracy_keras):.4f}")

models = ['Perceptron', 'Feed Forward NN']
accuracies = [accuracy, accuracy_keras]
```

```
plt.figure(figsize=(8, 6))
bars = plt.bar(models, accuracies, color=['skyblue', 'lightcoral'])
plt.ylabel('Accuracy')
plt.title('Model Accuracy Comparison')
plt.ylim(0.9, 1.0)

for bar, acc in zip(bars, accuracies):
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 0.001,
             f'{acc:.4f}', ha='center', va='bottom')

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

```
Model Accuracy Comparison:
Perceptron Accuracy: 0.9545
Keras Neural Network Accuracy: 0.9545
Difference: 0.0000
```

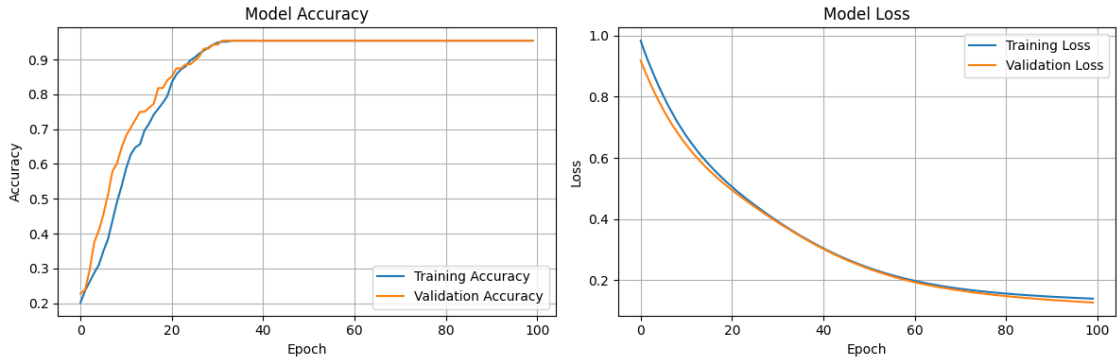


- Both Perceptron and Single Hidden Layer peak at 95.45% accuracy
- This shows that both perform just as good
- This likely is because there is no non-linearity in the data provided which eliminates the

purpose of need for hidden layer

- The data is linearly separable thus both models perform equally well

[14]:
```
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)

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



- As we see this is the required curve that we see
- As the numbe of epochs increase model accuracy increased and loss function decreses meaning the difference between what we predicted and actual value is not that much it gets closer to actual value this shows our model to be getting better each epoch
- We see after around 25 or so epoches the accuracy plateus