```
# Use the Magic Gamma Telescope database:
# 1. Apply an Artificial Neural Network (ANN) classifier with:
# • A(1-1-1) NN architecture.
# • A(1-2-1) NN architecture

!pip install numpy pandas tensorflow scikit-learn


import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow import keras
from tensorflow.keras import Input , layers
from sklearn.metrics import precision_score, recall_score, f1_score


data = pd.read_csv("C:/Users/HP/OneDrive/Desktop/ml 7th sem codes/datasets/magic.csv")
data


#ANN Classifier: 1-1-1
X = data.drop('class', axis=1).values  # Features
y = data["class"].apply(lambda x: 1 if x == "g" else 0)  # Target variable

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

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
```

```
X_test = scaler.transform(X_test)

#create a model model_a
model_a = keras.Sequential([
   layers.Input(shape=(X_train.shape[1],)),
   layers.Dense(1, activation='relu'),  # Hidden layer
   layers.Dense(1, activation='sigmoid')  # Output layer
])

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

model_a.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)



# Evaluate the model
from sklearn.metrics import precision_score, recall_score, f1_score

loss_A, accuracy_A = model_a.evaluate(X_test, y_test)

y_pred_A = (model_a.predict(X_test) > 0.5).astype("int32")

precision_A = precision_score(y_test, y_pred_A)
recall_A = recall_score(y_test, y_pred_A)
f1_A = f1_score(y_test, y_pred_A)

print(f"Model A - Loss: {loss_A}, Accuracy: {accuracy_A}")
print(f"Model A - Precision: {precision_A}, Recall: {recall_A}, F1 Score: {f1_A}")



#ANN 1-1-1-1
```

```
#create a model model_b
model_b = keras.Sequential([
   layers.Input(shape=(X_train.shape[1],)),
   layers.Dense(10, activation='relu'),# Hidden layer
   layers.Dense(5,activation='relu'),#Hidden layer
   layers.Dense(1, activation='sigmoid')  # Output layer
])

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

# Train the model
model_b.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)




# Evaluate the model
loss_B, accuracy_B = model_b.evaluate(X_test, y_test)

y_pred_B = (model_b.predict(X_test) > 0.5).astype("int32")

precision_B = precision_score(y_test, y_pred_B)
recall_B = recall_score(y_test, y_pred_B)
f1_B = f1_score(y_test, y_pred_B)

print(f"Model B - Loss: {loss_B}, Accuracy: {accuracy_B}")
print(f"Model B - Precision: {precision_B}, Recall: {recall_B}, F1 Score: {f1_B}")



results = pd.DataFrame({
```

```
    "Model": ["Model A (1-1-1)", "Model B (1-1-1-1)"],
    "Accuracy": [accuracy_A, accuracy_B],
    "Precision": [precision_A, precision_B],
    "Recall": [recall_A, recall_B],
    "F1 Score": [f1_A, f1_B]
})

# Display the table
print(results)
```