

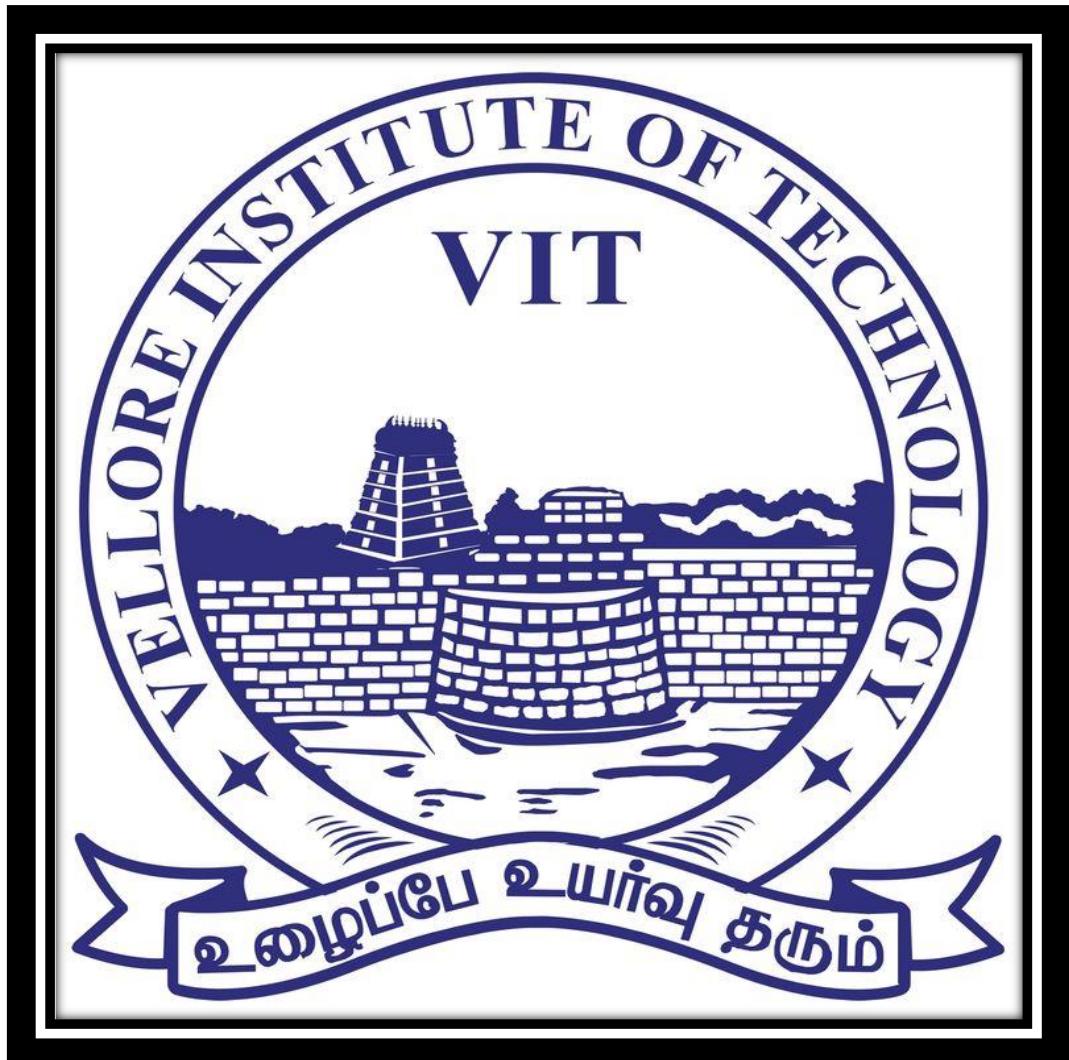


VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI



VELLORE INSTITUTE OF TECHNOLOGY

VIT – CHENNAI CAMPUS

DEEP LEARNING LAB

LAB ASSIGNMENT – 2

FACULTY : DR. HARINI. S



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

BCSE332P - DL Lab

Lab 2 - DL Lab Assignment

AUTO-TUNING OF PCAM DATASET:

CODE:

```
train_images, train_labels = load_hdf5_dataset(  
    "./pcam/pcam_dataset/camelyonpatch_level_2_split_train_x.h5",  
    "./pcam/pcam_dataset/camelyonpatch_level_2_split_train_y.h5"  
)  
  
test_images, test_labels = load_hdf5_dataset(  
    "./pcam/pcam_dataset/camelyonpatch_level_2_split_test_x.h5",  
    "./pcam/pcam_dataset/camelyonpatch_level_2_split_test_y.h5"  
)  
  
# Reshape labels  
train_labels = train_labels.reshape(-1, 1)  
train_labels.shape  
  
# Define the model creation function  
def create_model(optimizer='adam', dropout_rate=0.3):  
    model = Sequential([  
        Conv2D(32, (3, 3), activation='relu',  
        input_shape=(train_images.shape[1], train_images.shape[2],  
        train_images.shape[3])),  
        MaxPooling2D(2, 2),
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

Conv2D(64, (3, 3), activation='relu'),

MaxPooling2D(2, 2),

Conv2D(128, (3, 3), activation='relu'),

MaxPooling2D(2, 2),

Flatten(),

Dense(128, activation='relu'),

Dropout(dropout_rate),

Dense(1, activation='sigmoid')

)

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

```
return model
```

```
# Create KerasClassifier without specifying any custom parameters
```

```
keras_model = KerasClassifier(model=create_model, epochs=5,
batch_size=32, verbose=0, dropout_rate=0.3)
```

```
# Define the parameter grid for grid search
```

```
param_grid = {
```

```
'optimizer': ['adam', 'rmsprop'],
```

```
'dropout_rate': [0.3, 0.5]
```

```
}
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
subset_train_images = train_images[:1000]
subset_train_labels = train_labels[:1000]
subset_test_images = train_images[:300]
subset_test_labels = train_labels[:300]
grid_search = GridSearchCV(estimator=keras_model,
param_grid=param_grid, cv=3)
grid_search
grid_result = grid_search.fit(subset_train_images, subset_train_labels)
grid_result
# Print results
print("Best: %f using %s" % (grid_result.best_score_,
grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
from sklearn.model_selection import KFold
kfold = KFold(n_splits=5, shuffle=True, random_state=42)
cvscores = []
for train, test in kfold.split(subset_train_images, subset_train_labels):
    model = create_model()
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

```
model.compile(optimizer='rmsprop',
loss='categorical_crossentropy', metrics=['accuracy'])

model.fit(subset_train_images, subset_train_labels, epochs=2,
batch_size=32, verbose=0)

scores = model.evaluate(subset_test_images, subset_test_labels,
verbose=0)
```

```
print("Accuracy: %.2f%%" % (scores[1] * 100))

cvscores.append(scores[1] * 100)
```

```
print("Mean Accuracy: %.2f%% (+/- %.2f%%)" %
(np.mean(cvscores), np.std(cvscores)))
```

```
import h5py

import numpy as np

def load_hdf5_dataset(images_file_path, labels_file_path):

    with h5py.File(images_file_path, 'r') as f:

        images = np.array(f['x'])

    with h5py.File(labels_file_path, 'r') as f:

        labels = np.array(f['y'])

    return images, labels
```

```
train_labels = train_labels.reshape(-1, 1)
```



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

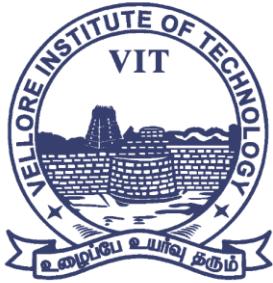
```
test_labels = test_labels.reshape(-1, 1)
```

```
import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
# Build the CNN model
```

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu',
           input_shape=(train_images.shape[1:])),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid') # Output layer for binary
                                 classification
])
```

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

```
# Train the model
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
model.fit(train_images, train_labels, epochs=3, batch_size=128)
```

```
# Evaluate performance on the test set
```

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
```

```
print('Test accuracy:', test_acc)
```

```
# Evaluate performance on the test set
```

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
```

```
print('Test accuracy:', test_acc)
```

OUTPUT OF THE CODE SNIPPET:

The screenshot shows a Jupyter Notebook interface running in a web browser. The notebook has a single cell containing Python code for loading an HDF5 dataset and evaluating a model. The code is as follows:

```
import h5py
import numpy as np
def load_hdf5_dataset(images_file_path, labels_file_path):
    with h5py.File(images_file_path, 'r') as f:
        images = np.array(f['x'])
    with h5py.File(labels_file_path, 'r') as f:
        labels = np.array(f['y'])
    return images, labels

train_images, train_labels = load_hdf5_dataset(
    r'/kaggle/input/metastatic-tissue-classification-patchcamelyon/pcam/training_split.h5',
    r'/kaggle/input/metastatic-tissue-classification-patchcamelyon/Labels/Labels/camelyonpatch_level_2_split.h5'
)

test_images, test_labels = load_hdf5_dataset(
    r'/kaggle/input/metastatic-tissue-classification-patchcamelyon/pcam/test_split.h5',
    r'/kaggle/input/metastatic-tissue-classification-patchcamelyon/Labels/Labels/camelyonpatch_level_2_split.h5'
)
```

To the right of the notebook, there is a sidebar with user profile information, including a profile picture of the user, their name "Vitta Vishnu Datta 21BAI1364", and their email "vittavishnu.datta2021@vitstudent.ac.in". There are also buttons for "Sync is on" and "Manage your Google Account". Below this, there is a "Data" section with a "+ Add" button, and a "Notebook" section showing a list of notebooks and files. The bottom of the screen shows a taskbar with various application icons.



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

21BAI1364 - PCAM Draft saved

[4]:

```
train_labels = train_labels.reshape(-1, 1)
test_labels = test_labels.reshape(-1, 1)
```

[6]:

```
import keras
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
# Build the CNN model
```

[8]:

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(train_images.shape[1:])),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid') # Output layer for binary classification
```

vitstudent.ac.in
Vitta Vishnu Datta 21BAI1364
vitvishnu.data2021@vitstudent.ac.in
Sync is on
Manage your Google Account
Other profiles
j (amma)
Vitta Vishnu (Vishnu Datta)
Guest
Add
test_metadata.csv
train_metadata.csv
valid_metadata.csv
camelyonpatch_Level_2_split_train_mask
camelyonpatch_Level_2_split_train_mask
pcam
Output
/kaggle/working

80°F Mostly clear 23:05 02-02-2024

21BAI1364 - PCAM Draft saved

[4]:

```
Conv2D(32, (3, 3), activation='relu', input_shape=(train_images.shape[1:])),
MaxPooling2D((2, 2)),
Conv2D(64, (3, 3), activation='relu'),
MaxPooling2D((2, 2)),
Flatten(),
Dense(64, activation='relu'),
Dense(1, activation='sigmoid') # Output layer for binary classification
```

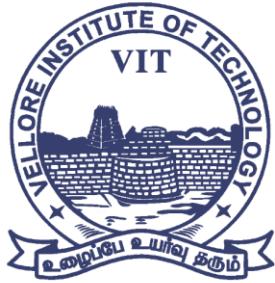
[10]:

```
# Train the model
model.fit(train_images, train_labels, epochs=3, batch_size=128)
# Evaluate performance on the test set
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)

Epoch 1/3
2048/2048 [=====] - 1691s 826ms/step - loss: 0.5921 - accuracy: 0.6716
Epoch 2/3
2048/2048 [=====] - 1667s 814ms/step - loss: 0.4925 - accuracy: 0.7677
Epoch 3/3
2048/2048 [=====] - 1718s 839ms/step - loss: 0.3917 - accuracy: 0.8268
1024/1024 [=====] - 35s 53ms/step - loss: 0.4932 - accuracy: 0.7727
Test accuracy: 0.772674569546875
```

vitstudent.ac.in
Vitta Vishnu Datta 21BAI1364
vitvishnu.data2021@vitstudent.ac.in
Sync is on
Manage your Google Account
Other profiles
j (amma)
Vitta Vishnu (Vishnu Datta)
Guest
Add
test_metadata.csv
train_metadata.csv
valid_metadata.csv
camelyonpatch_Level_2_split_train_mask
camelyonpatch_Level_2_split_train_mask
pcam
Output
/kaggle/working

80°F Mostly clear 23:06 02-02-2024



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Kaggle Notebook interface. The title bar says "21BAI1364 - PCAM Draft saved". The main area contains Python code for a machine learning model:

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)

Epoch 1/3
2048/2048 [=====] - 1691s 826ms/step - loss: 0.5921 - accuracy: 0.6716
Epoch 2/3
2048/2048 [=====] - 1667s 814ms/step - loss: 0.4925 - accuracy: 0.7677
Epoch 3/3
2048/2048 [=====] - 1718s 839ms/step - loss: 0.3917 - accuracy: 0.8268
1024/1024 [=====] - 55s 53ms/step - loss: 0.4932 - accuracy: 0.7727
Test accuracy: 0.772574560546875
```

Below the code, there's a cell with the same code and output:

```
# Evaluate performance on the test set
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)
```

The output shows:

```
1024/1024 [=====] - 56s 54ms/step - loss: 0.4932 - accuracy: 0.7727
Test accuracy: 0.772574560546875
```

The right sidebar shows user information for "vittavishnu.datta21BAI1364" and a file tree for the project:

- Data
- Input
 - model
 - camelyonpatch_level_2_split_train_mask
 - camelyonpatch_level_2_split_train_mask
 - pcam
- Output
 - /kaggle/working

PIMA DATASET:

CODE:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense
```

```
# Additional libraries for activation functions and optimizers
# (choose based on your needs)

from tensorflow.keras.layers import ReLU, Dropout
```



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

```
from tensorflow.keras.optimizers import Adam
```

```
# load Pima dataset
```

```
data = pd.read_csv('/kaggle/input/pima-indians-diabetes-database/diabetes.csv')
```

```
data.head()
```

```
# Separate features (X) and target variable (y)
```

```
X = data.drop('Outcome', axis=1)
```

```
y = data['Outcome']
```

```
# Split data into 80% training and 20% testing sets
```

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

```
# Scale features for better performance
```

```
scaler = StandardScaler()
```

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

```
X_test = scaler.transform(X_test)
```

```
# Define a simple Sequential model with 3 layers
```

```
model = Sequential([
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

```
Dense(12, activation='relu', input_shape=(X_train.shape[1],)), # 12 hidden neurons
```

```
Dense(8, activation='relu'), # 8 hidden neurons
```

```
Dense(1, activation='sigmoid'), # Output neuron for binary classification
```

```
])
```

```
# Choose different layers, activations, and hyperparameters like epochs and batch size for optimization
```

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

```
model.fit(X_train, y_train, epochs=50, batch_size=32)
```

```
import matplotlib.pyplot as plt
```

```
# Model accuracy
```

```
history = model.history.history
```

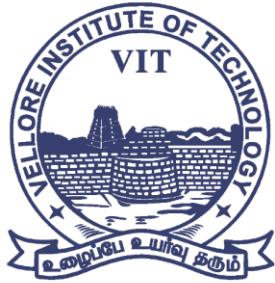
```
# Plot available metrics
```

```
plt.plot(history['accuracy']) # Plot training accuracy
```

```
# Check if validation accuracy is available:
```

```
if 'val_accuracy' in history:
```

```
plt.plot(history['val_accuracy']) # Plot validation accuracy
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'])
plt.show()
```

OUTPUT OF THE CODE SNIPPET:

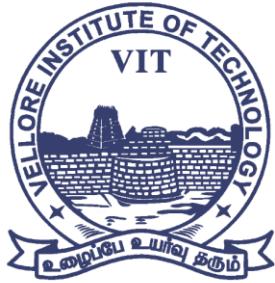
The screenshot shows a Kaggle notebook interface with several code cells. Cell [2] contains imports for pandas, sklearn, tensorflow, and keras. Cell [3] adds activation functions and optimizers. Cell [6] loads the Pima Indians dataset from a CSV file and prints its head. A sidebar on the right shows user profiles and recent datasets like 'Diabetes Dataset - Pima Indians' and 'Bayesian Logistic Regression wi...'. The status bar at the bottom indicates the system is running on a PC.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Additional libraries for activation functions and optimizers
# (choose based on your needs)
from tensorflow.keras.layers import ReLU, Dropout
from tensorflow.keras.optimizers import Adam

# load Pima dataset
data = pd.read_csv('/kaggle/input/pima-indians-diabetes-database/diabetes.csv')
data.head()

[6]: Pregnancies Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunction Age Outcome
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Kaggle notebook interface with the title "21BAI1364 - PIMA". The notebook contains the following Python code:

```
# load Pima dataset
data = pd.read_csv('/kaggle/input/pima-indians-diabetes-database/diabetes.csv')
data.head()

# Separate features (X) and target variable (y)
X = data.drop('Outcome', axis=1)
y = data['Outcome']

# Split data into 80% training and 20% testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale features for better performance
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Define a simple Sequential model with 3 layers
model = Sequential([
    Dense(12, activation='relu', input_shape=(X_train.shape[1],)), # 12 hidden neurons
    Dense(8, activation='relu'), # 8 hidden neurons
    Dense(1, activation='sigmoid'), # Output neuron for binary classification
])
```

A sidebar on the right shows a profile for "Vitta Vishnu Datta 21BAI1364" and a list of datasets including "Diabetes Dataset - Pima Indians" and "Bayesian Logistic Regression wi...".

The screenshot shows a Kaggle notebook interface with the title "21BAI1364 - PIMA". The notebook contains the same Python code as the previous screenshot:

```
# load Pima dataset
data = pd.read_csv('/kaggle/input/pima-indians-diabetes-database/diabetes.csv')
data.head()

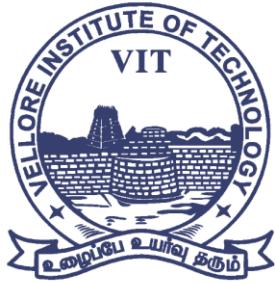
# Separate features (X) and target variable (y)
X = data.drop('Outcome', axis=1)
y = data['Outcome']

# Split data into 80% training and 20% testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale features for better performance
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Define a simple Sequential model with 3 layers
model = Sequential([
    Dense(12, activation='relu', input_shape=(X_train.shape[1],)), # 12 hidden neurons
    Dense(8, activation='relu'), # 8 hidden neurons
    Dense(1, activation='sigmoid'), # Output neuron for binary classification
])
```

A sidebar on the right shows a profile for "Vitta Vishnu Datta 21BAI1364" and a list of datasets including "Diabetes Dataset - Pima Indians" and "Bayesian Logistic Regression wi...".



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

kaggle.com/vittavishnudatta/21bai1364-pima/edit

21BAI1364 - PIMA Draft saved

File Edit View Run Add-ons Help

+ X 📄 🗂️ 🎨 🎵 Run All Code

Draft Session (5m)

Choose different layers, activations, and hyperparameters like epochs and batch size for optimization
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=[accuracy])
model.fit(X_train, y_train, epochs=50, batch_size=32)
import matplotlib.pyplot as plt

Epoch 1/50
20/20 [=====] - 1s 2ms/step - loss: 0.8195 - accuracy: 0.3583
Epoch 2/50
20/20 [=====] - 0s 2ms/step - loss: 0.7623 - accuracy: 0.4202
Epoch 3/50
20/20 [=====] - 0s 2ms/step - loss: 0.7218 - accuracy: 0.4739
Epoch 4/50
20/20 [=====] - 0s 2ms/step - loss: 0.6912 - accuracy: 0.5440
Epoch 5/50
20/20 [=====] - 0s 2ms/step - loss: 0.6682 - accuracy: 0.6205
Epoch 6/50
20/20 [=====] - 0s 2ms/step - loss: 0.6504 - accuracy: 0.6564
Epoch 7/50
20/20 [=====] - 0s 2ms/step - loss: 0.6340 - accuracy: 0.6743
Epoch 8/50
20/20 [=====] - 0s 2ms/step - loss: 0.6190 - accuracy: 0.6971
Epoch 9/50
20/20 [=====] - 0s 2ms/step - loss: 0.6036 - accuracy: 0.7085
Epoch 10/50
20/20 [=====] - 0s 2ms/step - loss: 0.5893 - accuracy: 0.7134
Epoch 11/50
20/20 [=====] - 0s 2ms/step - loss: 0.5758 - accuracy: 0.7199
Epoch 12/50

Kaggle uses cookies from Google to deliver and enhance the quality of its services and to analyze traffic.

Breaking news Unfolding now

Search

23:14 02-02-2024

kaggle.com/vittavishnudatta/21bai1364-pima/edit

21BAI1364 - PIMA Draft saved

File Edit View Run Add-ons Help

+ X 📄 🗂️ 🎨 🎵 Run All Code

Draft Session (6m)

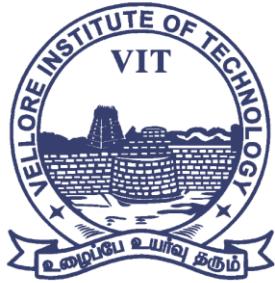
Epoch 1/50
20/20 [=====] - 0s 2ms/step - loss: 0.4493 - accuracy: 0.7736
Epoch 2/50
20/20 [=====] - 0s 2ms/step - loss: 0.4479 - accuracy: 0.7720
Epoch 3/50
20/20 [=====] - 0s 2ms/step - loss: 0.4467 - accuracy: 0.7769
Epoch 4/50
20/20 [=====] - 0s 2ms/step - loss: 0.4456 - accuracy: 0.7769
Epoch 5/50
20/20 [=====] - 0s 2ms/step - loss: 0.4440 - accuracy: 0.7785
Epoch 6/50
20/20 [=====] - 0s 2ms/step - loss: 0.4429 - accuracy: 0.7818
Epoch 7/50
20/20 [=====] - 0s 2ms/step - loss: 0.4418 - accuracy: 0.7834
Epoch 8/50
20/20 [=====] - 0s 2ms/step - loss: 0.4409 - accuracy: 0.7818
Epoch 9/50
20/20 [=====] - 0s 2ms/step - loss: 0.4409 - accuracy: 0.7818
Epoch 10/50
20/20 [=====] - 0s 2ms/step - loss: 0.4402 - accuracy: 0.7818
Epoch 11/50
20/20 [=====] - 0s 2ms/step - loss: 0.4400 - accuracy: 0.7818
Epoch 12/50
20/20 [=====] - 0s 2ms/step - loss: 0.4388 - accuracy: 0.7785
Epoch 13/50
20/20 [=====] - 0s 2ms/step - loss: 0.4384 - accuracy: 0.7769
Epoch 14/50
20/20 [=====] - 0s 2ms/step - loss: 0.4371 - accuracy: 0.7834
Epoch 15/50
20/20 [=====] - 0s 2ms/step - loss: 0.4366 - accuracy: 0.7785
Epoch 16/50
20/20 [=====] - 0s 2ms/step - loss: 0.4362 - accuracy: 0.7834
Epoch 17/50
20/20 [=====] - 0s 2ms/step - loss: 0.4352 - accuracy: 0.7850
Epoch 18/50
20/20 [=====] - 0s 2ms/step - loss: 0.4344 - accuracy: 0.7818
Epoch 19/50
20/20 [=====] - 0s 2ms/step - loss: 0.4337 - accuracy: 0.7866
Epoch 20/50
20/20 [=====] - 0s 2ms/step - loss: 0.4331 - accuracy: 0.7899

Kaggle uses cookies from Google to deliver and enhance the quality of its services and to analyze traffic.

Breaking news Unfolding now

Search

23:14 02-02-2024



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

Kaggle.com - VIT Student

21BAI1364 - PIMA Draft saved

File Edit View Run Add-ons Help

+ X ☐ 🗑️ 🗃️ ⟲ ⟳ Run All Code

Draft Session (6m)

Epoch 43/50
20/20 [=====] - 0s 2ms/step - loss: 0.4384 - accuracy: 0.7769
Epoch 44/50
20/20 [=====] - 0s 2ms/step - loss: 0.4371 - accuracy: 0.7834
Epoch 45/50
20/20 [=====] - 0s 2ms/step - loss: 0.4366 - accuracy: 0.7785
Epoch 46/50
20/20 [=====] - 0s 2ms/step - loss: 0.4362 - accuracy: 0.7834
Epoch 47/50
20/20 [=====] - 0s 2ms/step - loss: 0.4352 - accuracy: 0.7850
Epoch 48/50
20/20 [=====] - 0s 2ms/step - loss: 0.4344 - accuracy: 0.7818
Epoch 49/50
20/20 [=====] - 0s 2ms/step - loss: 0.4337 - accuracy: 0.7866
Epoch 50/50
20/20 [=====] - 0s 2ms/step - loss: 0.4331 - accuracy: 0.7899

Model accuracy
history = model.history.history

Plot available metrics
plt.plot(history['accuracy']) # Plot training accuracy

[14]: <matplotlib.lines.Line2D at 0x7800e97094e0>

Kaggle uses cookies from Google to deliver and enhance the quality of its services and to analyze traffic.

vitstudent.ac.in

Add Data

Vitta Vishnu Datta 21BAI1364
vitavishnu.data2021@vitstudent.ac.in

Sync is on

Manage your Google Account

Other profiles

j (emma)

Vitta Vishnu (Vishnu Datta)

Guest

Diabetes Dataset - Pima Indians

Ms. Nancy Al Aswad - Updated 2y ago
112 Upvotes - CSV - 9 kB

Bayesian Logistic Regression wi...

Updated 6y ago
91 Upvotes

Learn more. Ok, Got It.

23:14 02-02-2024

Kaggle.com - VIT Student

21BAI1364 - PIMA Draft saved

File Edit View Run Add-ons Help

+ X ☐ 🗑️ 🗃️ ⟲ ⟳ Run All Code

Draft Session (6m)

Plot available metrics
plt.plot(history['accuracy']) # Plot training accuracy

[14]: <matplotlib.lines.Line2D at 0x7800e97094e0>

Kaggle uses cookies from Google to deliver and enhance the quality of its services and to analyze traffic.

vitstudent.ac.in

Add Data

Vitta Vishnu Datta 21BAI1364
vitavishnu.data2021@vitstudent.ac.in

Sync is on

Manage your Google Account

Other profiles

j (emma)

Vitta Vishnu (Vishnu Datta)

Guest

Diabetes Dataset - Pima Indians

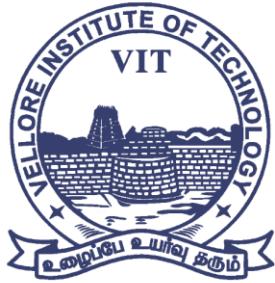
Ms. Nancy Al Aswad - Updated 2y ago
112 Upvotes - CSV - 9 kB

Bayesian Logistic Regression wi...

Updated 6y ago
91 Upvotes

Learn more. Ok, Got It.

23:15 02-02-2024



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
i 1: # Check if validation accuracy is available:  
if 'val_accuracy' in history:  
    plt.plot(history['val_accuracy']) # Plot validation accuracy  
plt.title('Model Accuracy')  
plt.ylabel('accuracy')  
plt.xlabel('epoch')  
plt.legend(['train', 'test'])  
plt.show()
```

The screenshot shows a Jupyter Notebook interface with a line plot titled "Model Accuracy". The x-axis is labeled "epoch" and ranges from 0 to 50. The y-axis is labeled "accuracy" and ranges from 0.4 to 0.7. The plot shows a single blue curve that starts at approximately (0, 0.35) and rises steeply to about (10, 0.75), then levels off. Below the plot is the corresponding Python code. To the right of the notebook, there is a sidebar for "vitsstudent.ac.in" showing user information and a list of datasets. At the bottom, there is a taskbar with various icons and system status.

AUTO-TUNING OF PIMA DATASET:

```
!pip3 install --upgrade tensorflow
```

```
!pip3 install scikeras[tensorflow]
```

```
# MLP for Pima Indians Dataset with 10-fold cross validation  
from keras.models import Sequential  
from keras.layers import Dense  
from sklearn.model_selection import StratifiedKFold  
import numpy  
# fix random seed for reproducibility  
seed = 7
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
numpy.random.seed(seed)
```

```
# load pima indians dataset
```

```
import pandas as pd
```

```
import numpy as np
```

```
diabetes_data = pd.read_csv('/kaggle/input/pima-indians-diabetes-database/diabetes.csv')
```

```
dataset = diabetes_data.to_numpy()
```

```
# split into input (X) and output (Y) variables
```

```
X = dataset[:,0:8]
```

```
Y = dataset[:,8]
```

```
# define 10-fold cross validation test harness
```

```
kfold = StratifiedKFold(n_splits=10, shuffle=True,  
random_state=seed)
```

```
cvscores = []
```

```
for train, test in kfold.split(X, Y):
```

```
    # create model
```

```
    model = Sequential()
```

```
    model.add(Dense(12, input_dim=8, kernel_initializer='uniform',  
activation= 'relu' ))
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

```
model.add(Dense(8, kernel_initializer='uniform' , activation=
"relu" ))  
  
model.add(Dense(1, kernel_initializer= 'uniform' , activation=
"sigmoid"))  
  
# Compile model  
  
model.compile(loss= "binary_crossentropy" , optimizer= "adam" ,
metrics=[ "accuracy" ])  
  
# Fit the model  
  
model.fit(X[train], Y[train], epochs=150, batch_size=10,
verbose=0) # evaluate the model  
  
scores = model.evaluate(X[test], Y[test], verbose=0)  
  
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))  
  
cvscores.append(scores[1] * 100)  
  
print("%.2f%% (+/- %.2f%%)" % (numpy.mean(cvscores),
numpy.std(cvscores)))
```

GRID SEARCH:

```
import numpy as np  
  
from sklearn import datasets  
  
from sklearn.model_selection import train_test_split, GridSearchCV  
  
from sklearn.svm import SVC  
  
from sklearn.metrics import accuracy_score
```



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

```
# Load the Pima Indians Diabetes dataset
```

```
pima_data = datasets.load_diabetes()
```

```
X = pima_data.data # Features
```

```
y = pima_data.target # Labels
```

```
# Split the dataset into training and testing sets
```

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

```
# Define the SVM model
```

```
svm_model = SVC()
```

```
# Define the parameter grid for grid search
```

```
param_grid = {
```

```
'C': [0.1, 1, 10, 100], # Wider range of C values for exploration
```

```
'kernel': ['linear', 'rbf'],
```

```
'gamma': ['scale', 'auto'] # Include 'scale' and 'auto' for gamma
```

```
}
```

```
# Initialize GridSearchCV
```

```
grid_search = GridSearchCV(svm_model, param_grid, cv=5,  
scoring='accuracy')
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# Fit the model to the data
```

```
grid_search.fit(X_train, y_train)
```

```
# Get the best hyperparameters
```

```
best_params = grid_search.best_params_
```

```
# Get the best model
```

```
best_model = grid_search.best_estimator_
```

```
# Make predictions on the test set using the best model
```

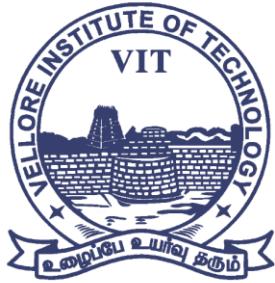
```
y_pred = best_model.predict(X_test)
```

```
# Calculate accuracy
```

```
accuracy = accuracy_score(y_test, y_pred)
```

```
print(f"Best Hyperparameters: {best_params}")
```

```
print(f"Test Accuracy: {accuracy}")
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

OUTPUT OF THE CODE SNIPPET:

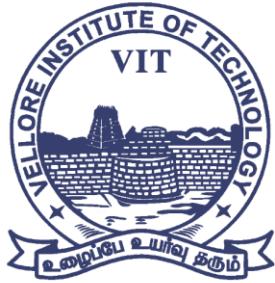
```
!pip3 install --upgrade tensorflow
!pip3 install scikeras[tensorflow]

Requirement already satisfied: tensorflow in /opt/conda/lib/python3.10/site-packages (2.15.0)
Requirement already satisfied: tensorflow<2.16,>=2.15.0 in /opt/conda/lib/python3.10/site-packages (from tensorflow)
WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('pip._vendor.urllib3.connection.HTTPSConnection object at 0x7aa9b0a32140: Failed to establish a new connection: [Errno -3] Temporary failure in name resolution')': /simple/tensorflow/
WARNING: Retrying (Retry(total=3, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('pip._vendor.urllib3.connection.HTTPSConnection object at 0x7aa9b0a32440: Failed to establish a new connection: [Errno -3] Temporary failure in name resolution')': /simple/tensorflow/
WARNING: Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('pip._vendor.urllib3.connection.HTTPSConnection object at 0x7aa9b0a325f0: Failed to establish a new connection: [Errno -3] Temporary failure in name resolution')': /simple/tensorflow/
WARNING: Retrying (Retry(total=1, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('pip._vendor.urllib3.connection.HTTPSConnection object at 0x7aa9b0a327a0: Failed to establish a new connection: [Errno -3] Temporary failure in name resolution')': /simple/tensorflow/
WARNING: Retrying (Retry(total=0, connect=None, read=None, redirect=None, status=None)) after connection broken by 'NewConnectionError('pip._vendor.urllib3.connection.HTTPSConnection object at 0x7aa9b0a32950: Failed to establish a new connection: [Errno -3] Temporary failure in name resolution')': /simple/tensorflow/
Requirement already satisfied: absl-py>=1.0.1 in /opt/conda/lib/python3.10/site-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /opt/conda/lib/python3.10/site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.2 in /opt/conda/lib/python3.10/site-packages (from tensorflow) (23.5.26)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /opt/conda/lib/python3.10/site-packages (from tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in /opt/conda/lib/python3.10/site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in /opt/conda/lib/python3.10/site-packages (from tensorflow) (3.10.0)
Requirement already satisfied: libclang>=13.0.0 in /opt/conda/lib/python3.10/site-packages (from tensorflow) (16.0.6)
```

Nasdaq 100 +1.78% Search Home Mail Calendar Dell WhatsApp Google Microsoft YouTube Chrome Edge Firefox 2318 02-02-2024

```
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in /opt/conda/lib/python3.10/site-packages (from tensorflow)
Requirement already satisfied: keras<2.16,>=2.15.0 in /opt/conda/lib/python3.10/site-packages (from tensorflow)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /opt/conda/lib/python3.10/site-packages (from astunparse>=1.6.0>tensorflow)
Requirement already satisfied: google-auth<3,>=1.6.3 in /opt/conda/lib/python3.10/site-packages (from tensorboard<2.16,>=2.15>tensorflow) (2.26.1)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /opt/conda/lib/python3.10/site-packages (from tensorboard<2.16,>=2.15>tensorflow) (1.2.0)
Requirement already satisfied: markdown>=2.6.8 in /opt/conda/lib/python3.10/site-packages (from tensorboard<2.16,>=2.15>tensorflow) (3.5.2)
Requirement already satisfied: requests<3,>=2.21.0 in /opt/conda/lib/python3.10/site-packages (from tensorboard<2.16,>=2.15>tensorflow) (2.31.1)
Requirement already satisfied: tensorboard-data-server<0.8,>=0.7.0 in /opt/conda/lib/python3.10/site-packages (from tensorboard<2.16,>=2.15>tensorflow) (0.7.2)
Requirement already satisfied: werkzeug<1.0.1 in /opt/conda/lib/python3.10/site-packages (from tensorboard<2.16,>=2.15>tensorflow) (3.0.1)
Requirement already satisfied: pyParsing>=3.0.5,>=2.0.2 in /opt/conda/lib/python3.10/site-packages (from packaging>tensorflow) (3.1.1)
Requirement already satisfied: cachetools>=6.0,>=2.0.0 in /opt/conda/lib/python3.10/site-packages (from google-auth<3,>=1.6.3>tensorflow)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /opt/conda/lib/python3.10/site-packages (from google-auth<3,>=1.6.3>tensorflow)
Requirement already satisfied: pyasn1>=0.2.1 in /opt/conda/lib/python3.10/site-packages (from google-auth<3,>=1.6.3>tensorflow)
Requirement already satisfied: rsa<5,>=3.1.4 in /opt/conda/lib/python3.10/site-packages (from google-auth<3,>=1.6.3>tensorflow)
Requirement already satisfied: requests>=2.21.0 in /opt/conda/lib/python3.10/site-packages (from requests<3,>=2.21.0>tensorflow)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/python3.10/site-packages (from requests<3,>=2.21.0>tensorflow)
Requirement already satisfied: idna4,>2.5 in /opt/conda/lib/python3.10/site-packages (from requests<3,>=2.21.0>tensorflow)
Requirement already satisfied: requests<3,>=2.21.0>tensorflow (3.6)
Requirement already satisfied: urllib3<3,>=2.1.1 in /opt/conda/lib/python3.10/site-packages (from requests<3,>=2.21.0>tensorflow)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.10/site-packages (from requests<3,>=2.21.0>tensorflow)
Requirement already satisfied: MarkupSafe>=2.1.1 in /opt/conda/lib/python3.10/site-packages (from werkzeug<1.0.1>tensorflow)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /opt/conda/lib/python3.10/site-packages (from pyasn1-modules>=0.2.1>google)
Requirement already satisfied: pyasn1>=0.4.6 in /opt/conda/lib/python3.10/site-packages (from pyasn1-modules>=0.2.1>google)
```

79°F Partly cloudy Search Home Mail Calendar Dell WhatsApp Google Microsoft YouTube Chrome Edge Firefox 2319 02-02-2024



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Kaggle notebook interface. The title bar says "21BAI1364 - PIMA Draft saved". The code editor contains the following Python script:

```
# MLP for Pima Indians Dataset with 10-fold cross validation
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import StratifiedKFold
import numpy
# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)
# load pima indians dataset
import pandas as pd
import numpy as np

diabetes_data = pd.read_csv('kaggle/input/pima-indians-diabetes-database/diabetes.csv')
dataset = diabetes_data.to_numpy()

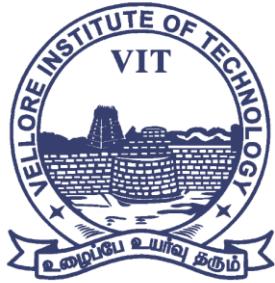
# split into input (X) and output (Y) variables
X = dataset[:,0:8]
Y = dataset[:,8]
# define 10-fold cross validation test harness
kfolds = StratifiedKFold(n_splits=10, shuffle=True, random_state=seed)
cvscores = []
for train, test in kfolds.split(X, Y):
    # create model
```

The status bar at the bottom right shows the date as 02-02-2024 and the time as 23:19.

The screenshot shows a Kaggle notebook interface. The title bar says "21BAI1364 - PIMA Draft saved". The code editor contains the following Python script:

```
kfolds = StratifiedKFold(n_splits=10, shuffle=True, random_state=seed)
cvscores = []
for train, test in kfolds.split(X, Y):
    # create model
    model = Sequential()
    model.add(Dense(12, input_dim=8, kernel_initializer='uniform', activation='relu'))
    model.add(Dense(8, kernel_initializer='uniform', activation='relu'))
    model.add(Dense(1, kernel_initializer='uniform', activation="sigmoid"))
    # Compile model
    model.compile(loss="binary_crossentropy", optimizers="adam", metrics=[ "accuracy"])
    # Fit the model
    model.fit(X[train], Y[train], epochs=150, batch_size=10, verbose=0) # evaluate the model
    scores = model.evaluate(X[test], Y[test], verbose=0)
    print("%s: %.2f%% (%s)" % (model.metrics_names[1], scores[1]*100))
    cvscores.append(scores[1] * 100)
print("%.2f%% (+/- %.2f%%)" % (numpy.mean(cvscores), numpy.std(cvscores)))
```

The status bar at the bottom right shows the date as 02-02-2024 and the time as 23:19.



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Jupyter Notebook interface with the following code:

```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load the Pima Indians Diabetes dataset
pima_data = datasets.load_diabetes()
X = pima_data.data # Features
y = pima_data.target # Labels

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

# Define the SVM model
svm_model = SVC()

# Define the parameter grid for grid search
param_grid = {
    'C': [0.1, 1, 10, 100], # Wider range of C values for exploration
    'kernel': ['linear', 'rbf'],
    'gamma': ['scale', 'auto'] # Include 'scale' and 'auto' for gamma
}
```

The notebook is titled "21BAI1364 - PIMA" and is in draft mode. A sidebar on the right shows user information and a list of datasets.

The screenshot continues the Jupyter Notebook code from the previous screenshot:

```
# Initialize GridSearchCV
grid_search = GridSearchCV(svm_model, param_grid, cv=5, scoring='accuracy')

# Fit the model to the data
grid_search.fit(X_train, y_train)

# Get the best hyperparameters
best_params = grid_search.best_params_

# Get the best model
best_model = grid_search.best_estimator_

# Make predictions on the test set using the best model
y_pred = best_model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

print(f"Best Hyperparameters: {best_params}")
print(f"Test Accuracy: {accuracy}")

/opt/conda/lib/python3.10/site-packages/sklearn/model_selection/_split.py:700: UserWarning: The least populated class in y has only 1 members, which is less than n_splits=5.
  warnings.warn(
Best Hyperparameters: {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}
Test Accuracy: 0.011235955056179775
```

The notebook is titled "21BAI1364 - PIMA" and is in draft mode. A sidebar on the right shows user information and a list of datasets. A warning message is visible at the bottom of the code cell.



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

AUTO-TUNING FOR MNIST DATASET:

CODE:

```
from keras.datasets import mnist
from keras.utils import to_categorical
from keras import models, layers
from sklearn.model_selection import KFold
import numpy as np
```

```
# Load the MNIST dataset
(train_images, train_labels), (test_images, test_labels) =
mnist.load_data()
```

```
# Flatten the images from 28x28 to a 1D array
train_images_flat = train_images.reshape((len(train_images), 28 * 28))
test_images_flat = test_images.reshape((len(test_images), 28 * 28))
# Normalize pixel values to be between 0 and 1
train_images_flat = train_images_flat.astype('float32') / 255
test_images_flat = test_images_flat.astype('float32') / 255
```

```
# One-hot encode the labels
train_labels_onehot = to_categorical(train_labels)
test_labels_onehot = to_categorical(test_labels)
```

```
# Define the KFold parameters
kfold = KFold(n_splits=5, shuffle=True, random_state=42)
```

```
# Initialize an empty list to store accuracy scores for each fold
cvscores = []
```

```
# Loop through the folds
for train, test in kfold.split(train_images_flat, train_labels_onehot):
    # Create the model
    model = models.Sequential()
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
model.add(layers.Dense(512, activation='relu',
input_shape=(28*28,)))

model.add(layers.Dense(10, activation='softmax'))

# Compile the model
model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
metrics=['accuracy'])

# Train the model on the current fold
model.fit(train_images_flat[train], train_labels_onehot[train],
epochs=10, batch_size=32, verbose=0)

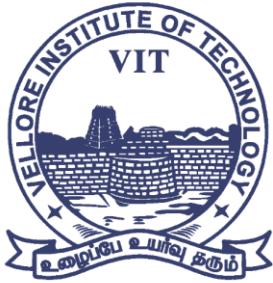
# Evaluate the model on the test set of the current fold
scores = model.evaluate(train_images_flat[test],
train_labels_onehot[test], verbose=0)

# Print and store the accuracy for the current fold
print("Accuracy: %.2f%%" % (scores[1] * 100))
cvscores.append(scores[1] * 100)
```

```
# Calculate and print the mean accuracy and standard deviation across
all folds
print("Mean Accuracy: %.2f%% (+/- %.2f%%)" % (np.mean(cvscores),
np.std(cvscores)))
# Define the model creation function
def create_model(optimizer='rmsprop', init='glorot_uniform'):
    model = Sequential()
    model.add(Dense(512, input_shape=(28 * 28,), activation='relu'))
    model.add(Dense(10, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer=optimizer,
metrics=['accuracy'])
    return model
```

```
# Set random seed for reproducibility
seed = 7
np.random.seed(seed)
```

```
# Create the Keras classifier
model1 = KerasClassifier(build_fn=create_model, verbose=0)
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
model1
# Define the hyperparameter grid

param_grid = {
    'optimizer': ['rmsprop', 'adam'],
    'epochs': [10, 20],
    'batch_size': [32, 64]
}
```

SCREENSHOTS OF THE CODE SNIPPET:

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.2.ipynb". The code cell contains the following Python script:

```
[14]: from keras.datasets import mnist
       from keras.utils import to_categorical
       from keras import models, layers
       from sklearn.model_selection import KFold
       import numpy as np

[15]: # load the MNIST dataset
       (train_images, train_labels), (test_images, test_labels) = mnist.load_data()
       Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
       11490434/11490434 [=====] - 0s 0us/step

[16]: # Flatten the images from 28x28 to a 1D array
       train_images_flat = train_images.reshape(len(train_images), 28 * 28)
       test_images_flat = test_images.reshape(len(test_images), 28 * 28)
       # Normalize pixel values to be between 0 and 1
       train_images_flat = train_images_flat.astype('float32') / 255
       test_images_flat = test_images_flat.astype('float32') / 255

[17]: # one-hot encode the labels
       train_labels_onehot = to_categorical(train_labels)
       test_labels_onehot = to_categorical(test_labels)

[18]: # define the KFold parameters
       kfold = KFold(n_splits=5, shuffle=True, random_state=42)
```

The notebook interface shows the code being run in a Jupyter-like environment. The status bar at the bottom indicates the disk usage and system information.



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.2.ipynb". The code in cell [20] initializes a Sequential model with two layers (512 and 10 units) and compiles it with RMSprop optimizer and categorical crossentropy loss. It then splits the data into training and testing sets, fits the model to the training set, evaluates it on the test set, and prints the accuracy for each fold. The output shows five folds with 97.92%, 98.15%, 97.95%, 97.81%, and 97.76% accuracy respectively.

```
[19] # Initialize an empty list to store accuracy scores for each fold
cvscores = []

[20] # Loop through the folds
for train, test in kfold.split(train_images_flat, train_labels_onehot):
    # Create the model
    model = models.Sequential()
    model.add(layers.Dense(512, activation='relu', input_shape=(28*28,)))
    model.add(layers.Dense(10, activation='softmax'))

    # Compile the model
    model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])

    # Train the model on the current fold
    model.fit(train_images_flat[train], train_labels_onehot[train], epochs=10, batch_size=32, verbose=0)

    # Evaluate the model on the test set of the current fold
    scores = model.evaluate(train_images_flat[test], train_labels_onehot[test], verbose=0)

    # Print and store the accuracy for the current fold
    print("Accuracy: %.2f%% (%s)" % (scores[1] * 100))
    cvscores.append(scores[1] * 100)

Accuracy: 97.92%
Accuracy: 98.15%
Accuracy: 97.95%
Accuracy: 97.81%
Accuracy: 97.76%
```

Cell [21] calculates the mean accuracy and standard deviation across all folds.

```
[21] # calculate and print the mean accuracy and standard deviation across all folds
print("Mean Accuracy: %.2f%% (%.2f%%)" % (np.mean(cvscores), np.std(cvscores)))
```

Cells [22] through [26] define a Keras classifier using the Sequential model, set a random seed, and define a hyperparameter grid.

```
[22] # set random seed for reproducibility
seed = 7
np.random.seed(seed)

[23] # Create the Keras classifier
model1 = KerasClassifier(build_fn=create_model, verbose=0)
model1

[24] # Define the hyperparameter grid
param_grid = [
    'optimizer': ['rmsprop', 'adam'],
    'epochs': [10, 20],
    'batch_size': [32, 64]
]
```

This screenshot shows the continuation of the Google Colab session. The code in cell [24] defines a hyperparameter grid for the Keras classifier. Cells [25] and [26] show the execution of the classifier and the grid search.

```
[25] # Fit the classifier to the training data
model1.fit(X_train, y_train, validation_data=(X_val, y_val))

[26] # Define the grid search
grid_search = GridSearchCV(estimator=model1, param_grid=param_grid, cv=5)
grid_search
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.2.ipynb". The code in cell [21] calculates mean accuracy and standard deviation across folds, resulting in 97.76%. Cell [22] sets a random seed for reproducibility. Cell [23] creates a Keras classifier using Sequential model, Dense layers, and categorical crossentropy loss. Cell [24] defines a hyperparameter grid for optimizer, epochs, and batch size. The notebook interface includes a sidebar with files and a status bar showing disk usage and system information.

```
# calculate and print the mean accuracy and standard deviation across all folds
print("Mean Accuracy: %.{2}f% (+/- %.{2}f%)".format(np.mean(csvscores), np.std(csvscores)))
# Define the model creation function
def create_model(optimizer='rmsprop', init='glorot_uniform'):
    model = Sequential()
    model.add(Dense(512, input_shape=(28 * 28,), activation='relu'))
    model.add(Dense(10, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
    return model

# Set random seed for reproducibility
seed = 7
np.random.seed(seed)

# Create the Keras classifier
model1 = KerasClassifier(build_fn=create_model, verbose=0)
model1

# Define the hyperparameter grid

param_grid = {
    'optimizer': ['rmsprop', 'adam'],
    'epochs': [10, 20],
    'batch_size': [32, 64]
}
```

AUTO-TUNING FOR FASHION MNIST DATASET:

CODE:

```
# Load the Fashion MNIST dataset
(train_images, train_labels), (test_images, test_labels) =
fashion_mnist.load_data()
```

```
# Flatten the images from 28x28 to a 1D array
train_images_flat = train_images.reshape((len(train_images), 28 * 28))
test_images_flat = test_images.reshape((len(test_images), 28 * 28))
# Normalize pixel values to be between 0 and 1
train_images_flat = train_images_flat.astype('float32') / 255
test_images_flat = test_images_flat.astype('float32') / 255
```

```
# One-hot encode the labels
train_labels_onehot = to_categorical(train_labels)
test_labels_onehot = to_categorical(test_labels)
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
# Define the KFold parameters
kfold = KFold(n_splits=5, shuffle=True, random_state=42)

# Initialize an empty list to store accuracy scores for each fold
cvscores = []

# Loop through the folds
for train, test in kfold.split(train_images_flat, train_labels_onehot):
    # Create the model
    model = models.Sequential()
    model.add(layers.Dense(512, activation='relu',
input_shape=(28*28,)))
    model.add(layers.Dense(10, activation='softmax'))

    # Compile the model
    model.compile(optimizer='rmsprop', loss='categorical_crossentropy',
metrics=['accuracy'])

    # Train the model on the current fold
    model.fit(train_images_flat[train], train_labels_onehot[train],
epochs=10, batch_size=32, verbose=0)

    # Evaluate the model on the test set of the current fold
    scores = model.evaluate(train_images_flat[test],
train_labels_onehot[test], verbose=0)

    # Print and store the accuracy for the current fold
    print("Accuracy: %.2f%%" % (scores[1] * 100))
    cvscores.append(scores[1] * 100)
```

```
from sklearn.model_selection import GridSearchCV
```

```
# Calculate and print the mean accuracy and standard deviation across
all folds
print("Mean Accuracy: %.2f%% (+/- %.2f%%)" % (np.mean(cvscores),
np.std(cvscores)))
```

```
# Define the model creation function
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
def create_model(optimizer='rmsprop', init='glorot_uniform'):
    model = Sequential()
    model.add(Dense(512, input_shape=(28 * 28,), activation='relu'))
    model.add(Dense(10, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer=optimizer,
metrics=['accuracy'])
    return model
```

```
# Set random seed for reproducibility
seed = 7
np.random.seed(seed)
```

```
# Create the Keras classifier
model2 = KerasClassifier(build_fn=create_model, verbose=0)
model2
```

```
# Define the hyperparameter grid

param_grid = {
    'optimizer': ['rmsprop', 'adam'],
    'epochs': [10, 20],
    'batch_size': [32, 64]
}
```

```
import h5py
import numpy as np
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

def load_hdf5_dataset(images_file_path, labels_file_path):
    with h5py.File(images_file_path, 'r') as f:
        images = np.array(f['x'])
    with h5py.File(labels_file_path, 'r') as f:
        labels = np.array(f['y'])
    return images, labels
```



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

SCREENSHOTS OF THE CODE SNIPPET:

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN2.ipynb". The code cell [24] contains the following Python code to load the Fashion MNIST dataset:

```
# Load the Fashion MNIST dataset
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

Cells [25] and [26] show code to flatten the images and normalize pixel values:

```
# Flatten the images from 28x28 to a 1D array
train_images_flat = train_images.reshape(len(train_images), 28 * 28)
test_images_flat = test_images.reshape(len(test_images), 28 * 28)

# Normalize pixel values to be between 0 and 1
train_images_flat = train_images_flat.astype('float32') / 255
test_images_flat = test_images_flat.astype('float32') / 255
```

Cell [27] defines KFold parameters and initializes an empty list for accuracy scores:

```
# Define the KFold parameters
kfold = KFold(n_splits=5, shuffle=True, random_state=42)

# Initialize an empty list to store accuracy scores for each fold
cvscores = []
```

The status bar at the bottom indicates "79°F Partly cloudy" and the date "02-02-2024".

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN2.ipynb". The code cell [27] defines KFold parameters and initializes an empty list for accuracy scores:

```
# Define the KFold parameters
kfold = KFold(n_splits=5, shuffle=True, random_state=42)

# Initialize an empty list to store accuracy scores for each fold
cvscores = []
```

Cell [28] contains the main loop for KFold cross-validation:

```
# Loop through the folds
for train, test in kfold.split(train_images_flat, train_labels_onehot):
    # Create the model
    model = models.Sequential()
    model.add(layers.Dense(512, activation='relu', input_shape=(28*28,)))
    model.add(layers.Dense(10, activation='softmax'))

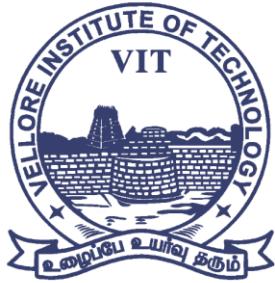
    # Compile the model
    model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])

    # Train the model on the current fold
    model.fit(train_images_flat[train], train_labels_onehot[train], epochs=10, batch_size=32, verbose=0)

    # Evaluate the model on the test set of the current fold
    scores = model.evaluate(train_images_flat[test], train_labels_onehot[test], verbose=0)

    # Print and store the accuracy for the current fold
    print("Accuracy: %.2f%%" % (scores[1] * 100))
    cvscores.append(scores[1] * 100)
```

The status bar at the bottom indicates "Breaking news Unfolding now" and the date "02-02-2024".



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.2.ipynb". The code in cell [28] prints accuracy values for multiple runs. Cells [29] and [30] import GridSearchCV and calculate mean accuracy. Cell [31] defines a model creation function. Cell [32] sets a random seed. The code in cell [33] creates a Keras classifier using the defined function. A tooltip for the KerasClassifier object in cell [33] is visible, showing its constructor parameters. Cell [34] defines a hyperparameter grid.

```
print("Accuracy: %.2f%%" % (scores[1] * 100))
cvscores.append(scores[1] * 100)

[29] from sklearn.model_selection import GridSearchCV
[30] # Calculate and print the mean accuracy and standard deviation across all folds
print("Mean Accuracy: %.2f%% (+/- %.2f%%)" % (np.mean(cvscores), np.std(cvscores)))

Mean Accuracy: 87.88% (+/- 0.47%)

[31] # Define the model creation function
def create_model(optimizer='rmsprop', init='glorot_uniform'):
    model = Sequential()
    model.add(Dense(512, input_shape=(28 * 28,), activation='relu'))
    model.add(Dense(10, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
    return model

[32] # Set random seed for reproducibility
seed = 7
np.random.seed(seed)

[33] # Create the Keras classifier
model2 = KerasClassifier(build_fn=create_model, verbose=0)
model2

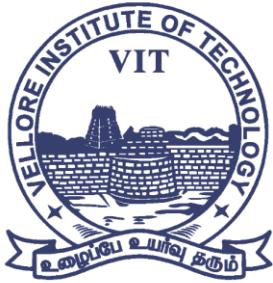
[34] # Define the hyperparameter grid
param_grid = {
    'optimizer': ['rmsprop', 'adam'],
    'epochs': [10, 20],
    'batch_size': [32, 64]
}
```

The screenshot continues the Google Colab notebook. The code in cell [33] creates a Keras classifier using the defined function. A tooltip for the KerasClassifier object in cell [33] is visible, showing its constructor parameters. Cell [34] defines a hyperparameter grid.

```
# Set random seed for reproducibility
seed = 7
np.random.seed(seed)

# Create the Keras classifier
model2 = KerasClassifier(build_fn=create_model, verbose=0)
model2

# Define the hyperparameter grid
param_grid = {
    'optimizer': ['rmsprop', 'adam'],
    'epochs': [10, 20],
    'batch_size': [32, 64]
}
```



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

AUTO-TUNING FOR DOGS&CATS CLASSIFICATION:

CODE:

```
(train_images, train_labels), (test_images, test_labels) =  
tf.keras.datasets.cifar10.load_data()  
train_images.shape
```

```
print(len(train_labels))  
train_labels  
test_images.shape  
len(test_labels)  
test_labels
```

```
train_filter = np.where((train_labels == 3) | (train_labels == 5))[0]  
test_filter = np.where((test_labels == 3) | (test_labels == 5))[0]  
train_images, train_labels = train_images[train_filter],  
train_labels[train_filter]  
test_images, test_labels = test_images[test_filter],  
test_labels[test_filter]
```

```
# Change labels to binary (0 for cat, 1 for dog)  
train_labels = (train_labels == 5).astype(int).flatten()  
test_labels = (test_labels == 5).astype(int).flatten()
```

```
# Define the KFold parameters  
kfold = KFold(n_splits=5, shuffle=True, random_state=42)  
  
# Initialize an empty list to store accuracy scores for each fold  
cvscores = []
```

```
# Loop through the folds  
for train, test in kfold.split(train_images, train_labels):  
    # Create the model  
    dogs_v_cats_model = models.Sequential()
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
dogs_v_cats_model.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(32, 32, 3)))
dogs_v_cats_model.add(layers.MaxPooling2D((2, 2)))
dogs_v_cats_model.add(layers.Conv2D(64, (3, 3), activation='relu'))
dogs_v_cats_model.add(layers.MaxPooling2D((2, 2)))
dogs_v_cats_model.add(layers.Conv2D(64, (3, 3), activation='relu'))
dogs_v_cats_model.add(layers.Flatten())
dogs_v_cats_model.add(layers.Dense(64, activation='relu'))
dogs_v_cats_model.add(layers.Dense(1, activation='sigmoid'))

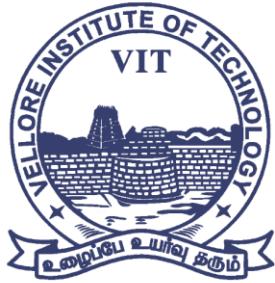
# Compile the model
dogs_v_cats_model.compile(optimizer='rmsprop',
loss='binary_crossentropy', metrics=['accuracy'])

# Train the model on the current fold
dogs_v_cats_model.fit(train_images[train], train_labels[train],
epochs=5, batch_size=32, verbose=0)

# Evaluate the model on the test set of the current fold
scores = dogs_v_cats_model.evaluate(train_images[test],
train_labels[test], verbose=0)

# Print and store the accuracy for the current fold
print("Accuracy: %.2f%%" % (scores[1] * 100))
cvscores.append(scores[1] * 100)
# Calculate and print the mean accuracy and standard deviation
across all folds
print("Mean Accuracy: %.2f%% (+/- %.2f%%)" % (np.mean(cvscores),
np.std(cvscores)))
```

```
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
def create_dogs_v_cats_model(optimizer='rmsprop'):
    cat_model = Sequential()
    cat_model.add(Conv2D(32, (3, 3), activation='relu',
input_shape=(32, 32, 3)))
    cat_model.add(MaxPooling2D((2, 2)))
    cat_model.add(Conv2D(64, (3, 3), activation='relu'))
    cat_model.add(MaxPooling2D((2, 2)))
    cat_model.add(Conv2D(64, (3, 3), activation='relu'))
    cat_model.add(Flatten())
    cat_model.add(Dense(64, activation='relu'))
```



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
cat_model.add(Dense(1, activation='sigmoid'))
cat_model.compile(optimizer=optimizer, loss='binary_crossentropy',
metrics=['accuracy'])
return cat_model

# Create the Keras classifier
dogs_v_cats_model = KerasClassifier(build_fn=create_dogs_v_cats_model,
epochs=5, batch_size=32, verbose=0)
```

SCREENSHOTS OF THE CODE SNIPPET:

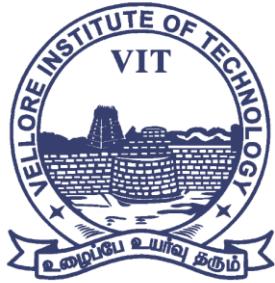
The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE32P LAB ASSIGN2.ipynb". The code cell contains Python code for loading CIFAR-10 data and filtering it to only include images of dogs and cats. The output shows the resulting arrays for training and testing images and labels.

```
(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.cifar10.load_data()
train_images.shape

[36] print(len(train_labels))
train_labels
test_images.shape
len(test_labels)
test_labels

50000
array([[1], [8], [8], ..., [5], [1], [7]], dtype=uint8)

[37] train_filter = np.where((train_labels == 3) | (train_labels == 5))[0]
test_filter = np.where((test_labels == 3) | (test_labels == 5))[0]
train_images, train_labels = train_images[train_filter], train_labels[train_filter]
test_images, test_labels = test_images[test_filter], test_labels[test_filter]
```



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

The screenshot shows a Google Colab notebook titled "21BAI1364 BCSE332P LAB ASSIGN.2.ipynb". The code in the notebook is as follows:

```
[37] train_filter = np.where((train_labels == 3) | (train_labels == 5))[0]
test_filter = np.where((test_labels == 3) | (test_labels == 5))[0]
train_images, train_labels = train_images[train_filter], train_labels[train_filter]
test_images, test_labels = test_images[test_filter], test_labels[test_filter]

[38] # Change labels to binary (0 for cat, 1 for dog)
train_labels = (train_labels == 5).astype(int).flatten()
test_labels = (test_labels == 5).astype(int).flatten()

[39] # Define the KFold parameters
kfolds = KFold(n_splits=5, shuffle=True, random_state=42)

# Initialize an empty list to store accuracy scores for each fold
cvscores = []

[40] # Loop through the folds
for train, test in kfolds.split(train_images, train_labels):
    # Create the model
    dogs_v_cats_model = models.Sequential()
    dogs_v_cats_model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
    dogs_v_cats_model.add(layers.MaxPooling2D((2, 2)))
    dogs_v_cats_model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    dogs_v_cats_model.add(layers.MaxPooling2D((2, 2)))
    dogs_v_cats_model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    dogs_v_cats_model.add(layers.Flatten())
    dogs_v_cats_model.add(layers.Dense(64, activation='relu'))
    dogs_v_cats_model.add(layers.Dense(1, activation='sigmoid'))
```

The notebook interface includes tabs for "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". A sidebar on the left shows a file tree with "sample_data". The status bar at the bottom indicates "Nasdaq 100 +1.50%" and "Disk 81.02 GB available".

The screenshot shows the continuation of the Google Colab notebook "21BAI1364 BCSE332P LAB ASSIGN.2.ipynb". The code in the notebook is as follows:

```
[40] # Loop through the folds
for train, test in kfolds.split(train_images, train_labels):
    # Create the model
    dogs_v_cats_model = models.Sequential()
    dogs_v_cats_model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
    dogs_v_cats_model.add(layers.MaxPooling2D((2, 2)))
    dogs_v_cats_model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    dogs_v_cats_model.add(layers.MaxPooling2D((2, 2)))
    dogs_v_cats_model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    dogs_v_cats_model.add(layers.Flatten())
    dogs_v_cats_model.add(layers.Dense(64, activation='relu'))
    dogs_v_cats_model.add(layers.Dense(1, activation='sigmoid'))

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

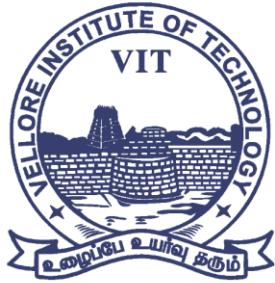
    # Train the model on the current fold
    dogs_v_cats_model.fit(train_images[train], train_labels[train], epochs=5, batch_size=32, verbose=0)

    # Evaluate the model on the test set of the current fold
    scores = dogs_v_cats_model.evaluate(train_images[test], train_labels[test], verbose=0)

    # Print and store the accuracy for the current fold
    print("Accuracy: %.2f%%" % (scores[1] * 100))
    cvscores.append(scores[1] * 100)

# Calculate and print the mean accuracy and standard deviation across all folds
print("Mean Accuracy: %.2f%% (+/- %.2f%%)" % (np.mean(cvscores), np.std(cvscores)))
```

The notebook interface includes tabs for "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". A sidebar on the left shows a file tree with "sample_data". The status bar at the bottom indicates "Nasdaq 100 +1.50%" and "Disk 81.02 GB available".



VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

```
File Edit View Insert Runtime Tools Help All changes saved
[40] dogs_v_cats_model.add(layers.Conv2D(64, (3, 3), activation='relu'))
dogs_v_cats_model.add(layers.MaxPooling2D((2, 2)))
dogs_v_cats_model.add(layers.Conv2D(64, (3, 3), activation='relu'))
dogs_v_cats_model.add(layers.Flatten())
dogs_v_cats_model.add(layers.Dense(64, activation='relu'))
dogs_v_cats_model.add(layers.Dense(1, activation='sigmoid'))

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

# Train the model on the current fold
dogs_v_cats_model.fit(train_images[train], train_labels[train], epochs=5, batch_size=32, verbose=0)

# Evaluate the model on the test set of the current fold
scores = dogs_v_cats_model.evaluate(train_images[test], train_labels[test], verbose=0)

# Print and store the accuracy for the current fold
print("Accuracy: %.2f%% (%.2f%%)" % (scores[1] * 100))
cvsscores.append(scores[1] * 100)
# Calculate and print the mean accuracy and standard deviation across all folds
print("Mean Accuracy: %.2f%% (+/- %.2f%%)" % (np.mean(cvsscores), np.std(cvsscores)))

Accuracy: 67.55%
Mean Accuracy: 67.55% (+/- 0.00%)
Accuracy: 64.86%
Mean Accuracy: 66.17% (+/- 1.37%)
Accuracy: 67.65%
Mean Accuracy: 66.67% (+/- 1.32%)
Accuracy: 67.20%
Mean Accuracy: 66.80% (+/- 1.17%)
Accuracy: 64.25%
Mean Accuracy: 66.29% (+/- 1.46%)

https://accounts.google.com/SignOutOptions?hl=en&continue=https://colab.research.google.com/drive/1jnV-0Dehc-zKzgShW4ZPDdVK4waE-K#ec=G8RAqQM e backend
Nasdaq 100 +1.50% Search File Explorer Home Mail Task View Taskbar 23:35 ENG IN 02-02-2024
```

```
File Edit View Insert Runtime Tools Help All changes saved
[40] cvsscores.append(scores[1] * 100)
# Calculate and print the mean accuracy and standard deviation across all folds
print("Mean Accuracy: %.2f%% (+/- %.2f%%)" % (np.mean(cvsscores), np.std(cvsscores)))

Accuracy: 67.55%
Mean Accuracy: 67.55% (+/- 0.00%)
Accuracy: 64.86%
Mean Accuracy: 66.17% (+/- 1.37%)
Accuracy: 67.65%
Mean Accuracy: 66.67% (+/- 1.32%)
Accuracy: 67.20%
Mean Accuracy: 66.80% (+/- 1.17%)
Accuracy: 64.25%
Mean Accuracy: 66.29% (+/- 1.46%)

[41] from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
def create_dogs_v_cats_model(optimizer='rmsprop'):
    cat_model = Sequential()
    cat_model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
    cat_model.add(MaxPooling2D((2, 2)))
    cat_model.add(Conv2D(64, (3, 3), activation='relu'))
    cat_model.add(MaxPooling2D((2, 2)))
    cat_model.add(Conv2D(64, (3, 3), activation='relu'))
    cat_model.add(Flatten())
    cat_model.add(Dense(64, activation='relu'))
    cat_model.add(Dense(1, activation='sigmoid'))
    cat_model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
    return cat_model

# Create the Keras classifier
dogs_v_cats_model = KerasClassifier(build_fn=create_dogs_v_cats_model, epochs=5, batch_size=32, verbose=0)

https://accounts.google.com/SignOutOptions?hl=en&continue=https://colab.research.google.com/drive/1jnV-0Dehc-zKzgShW4ZPDdVK4waE-K#ec=G8RAqQM e backend
Nasdaq 100 +1.50% Search File Explorer Home Mail Task View Taskbar 23:36 ENG IN 02-02-2024
```



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI



PREPARED AND SUBMITTED BY:

VITTA VISHNU DATTA.

21BAI1364 – SCOPE.

VIT – CHENNAI CAMPUS.