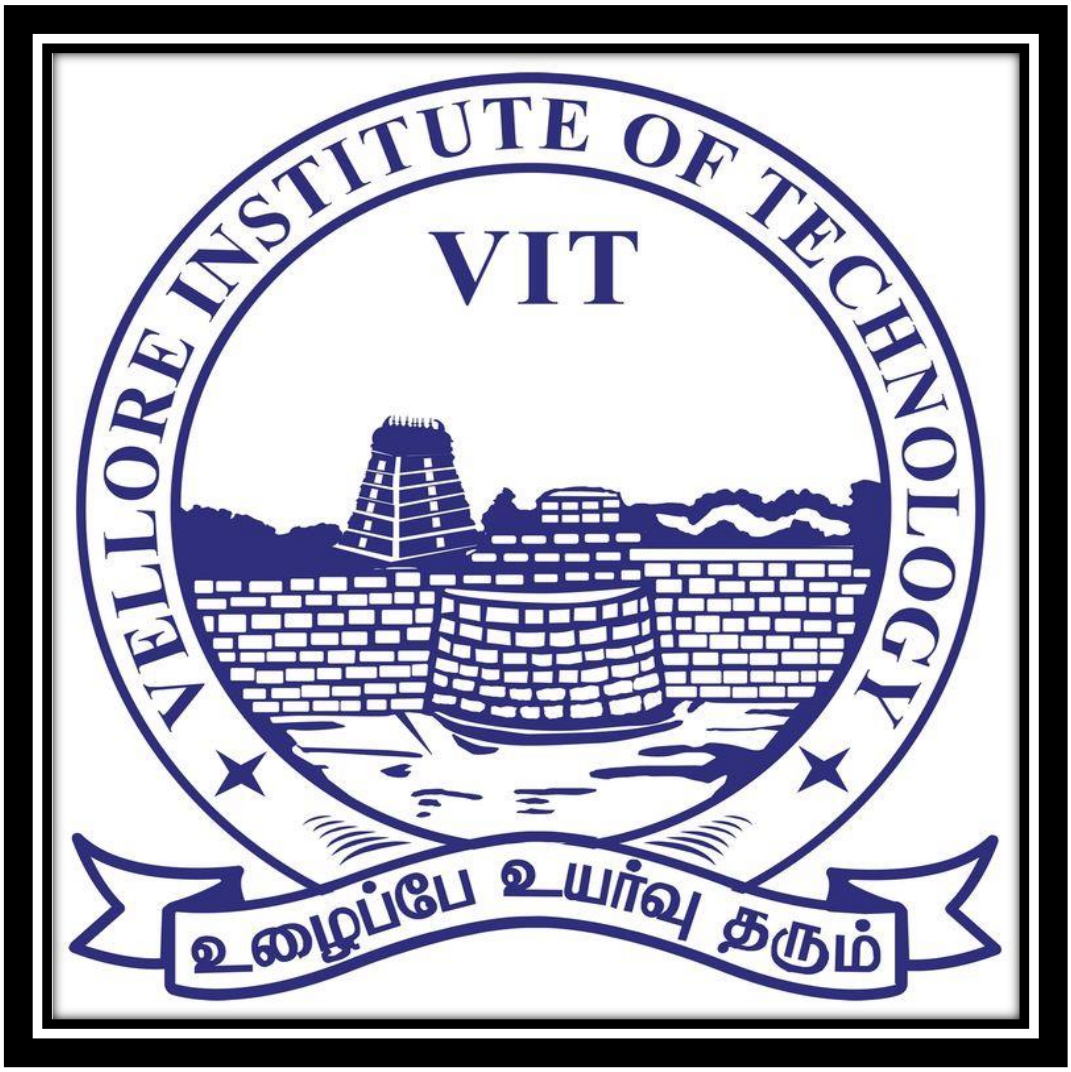
VIT®

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),
```

```
        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]
}
```

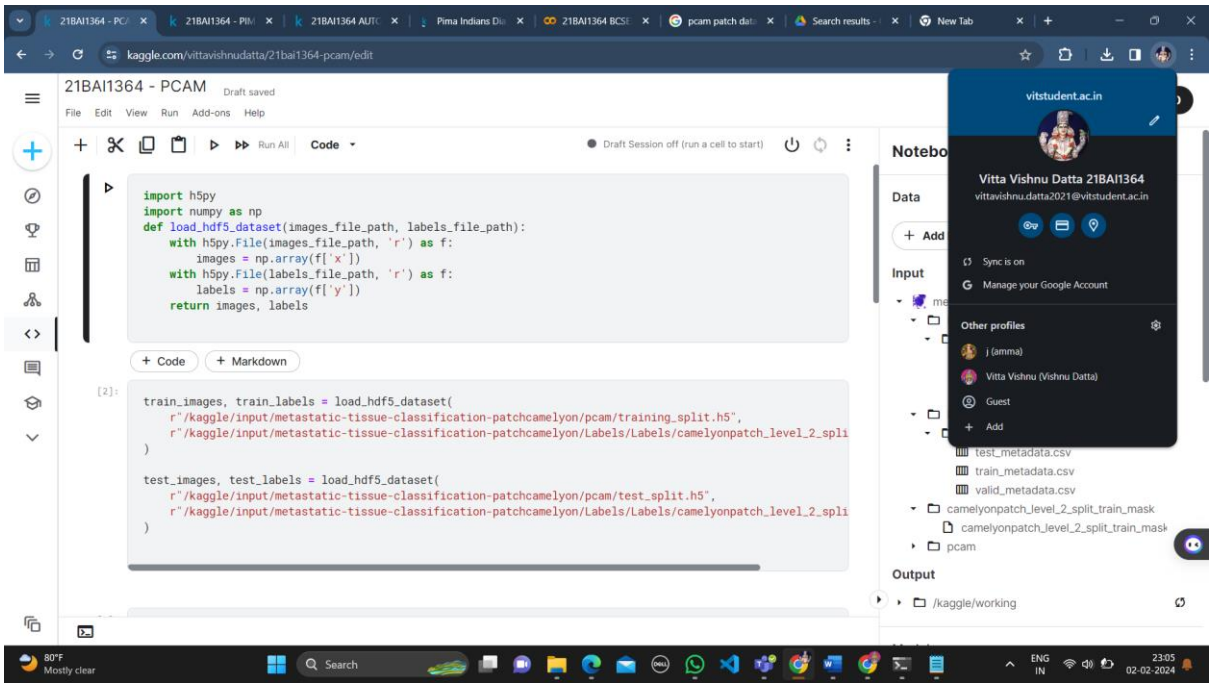
```
subset_train_images = train_images[:1000]

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()
```

```
    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)
```

```
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
```

```
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)
```

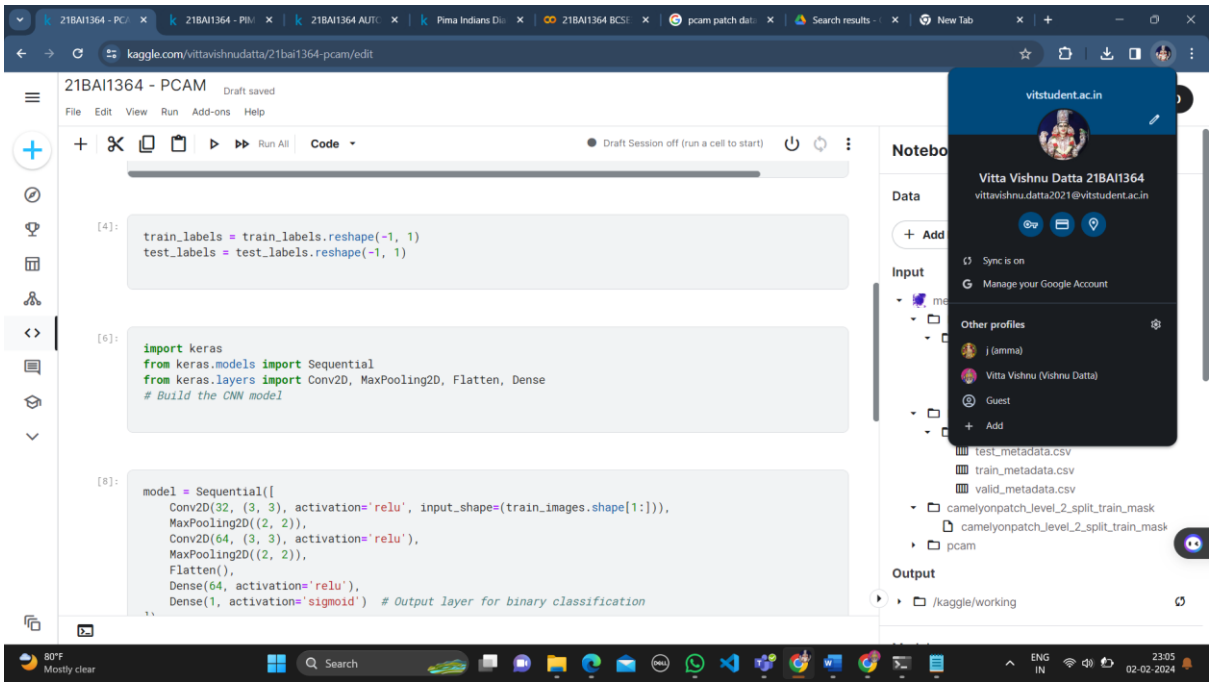
OUTPUT OF THE CODE SNIPPET:

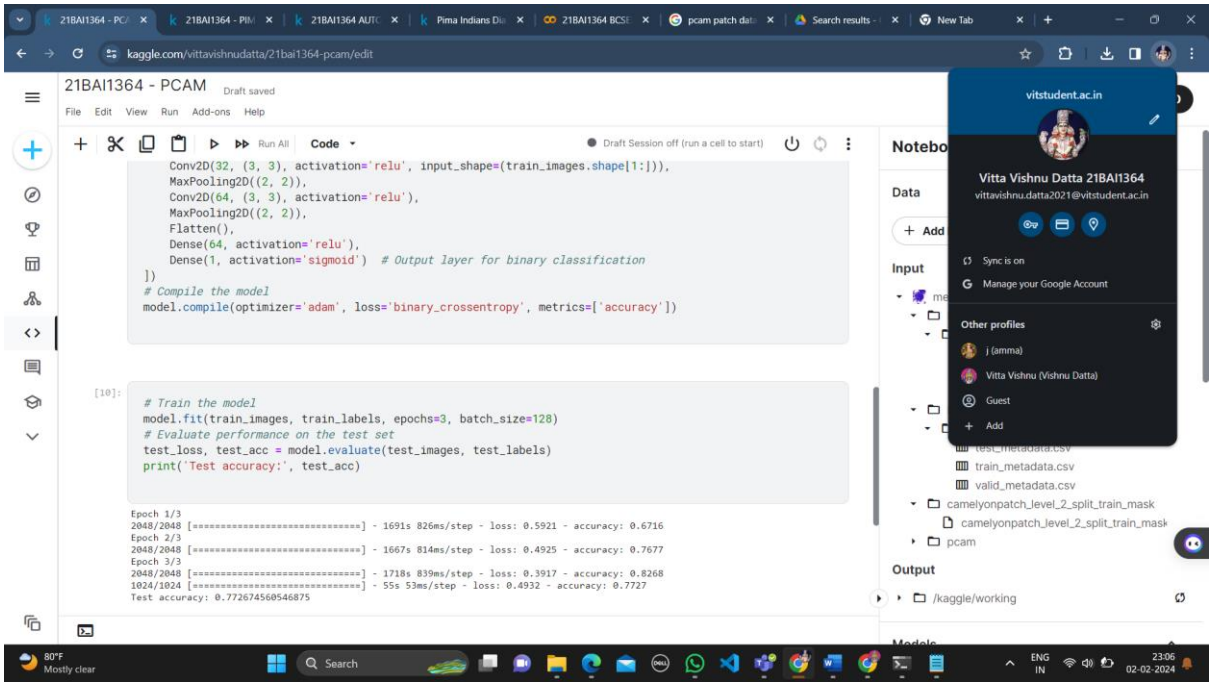


Vellore Institute of Technology

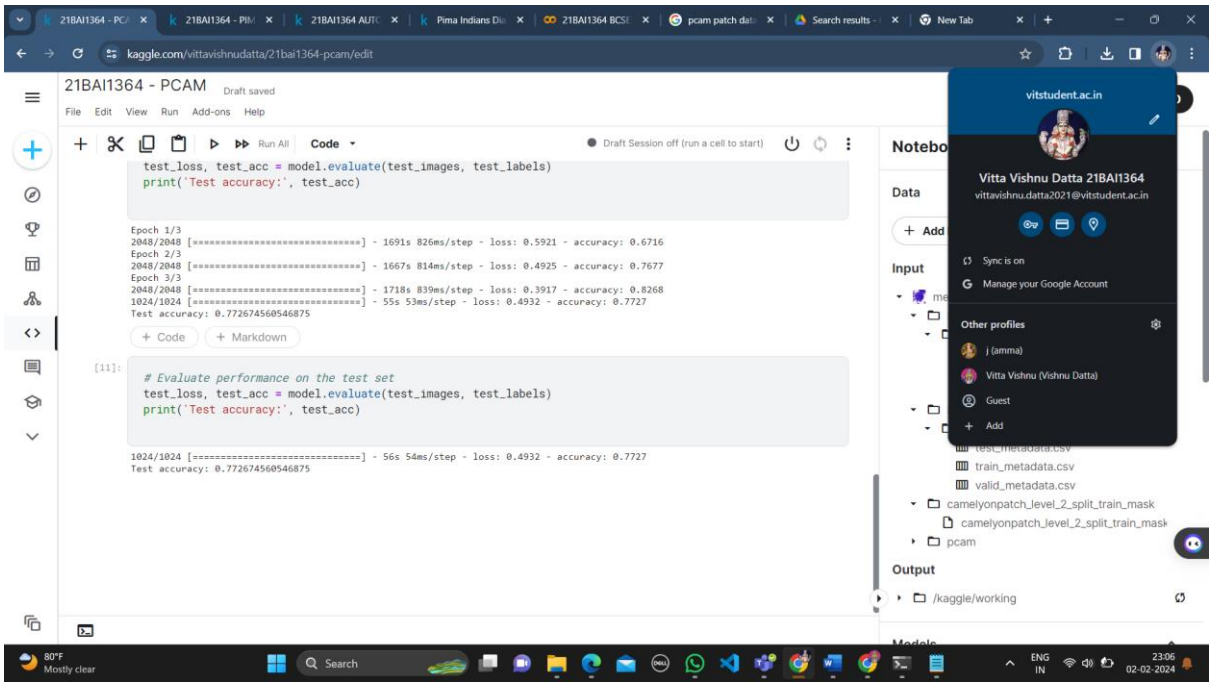
(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI





VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI



PIMA DATASET:

CODE:

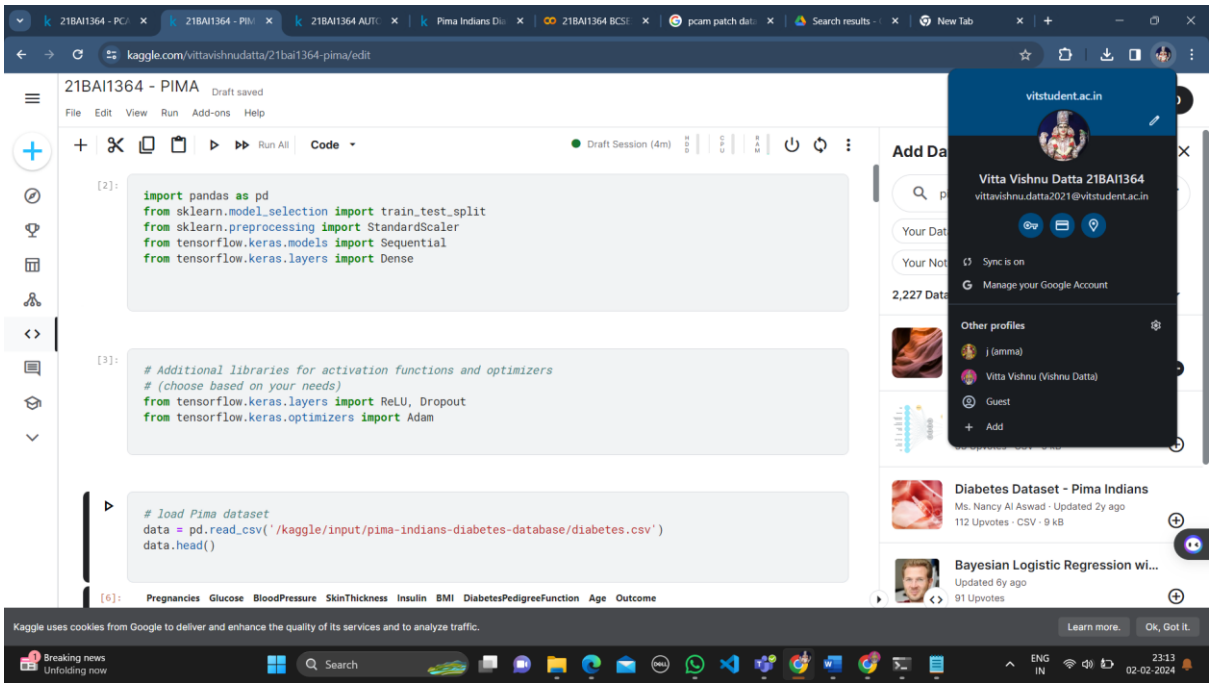
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

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

```
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([
```

```
    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
```

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

OUTPUT OF THE CODE SNIPPET:

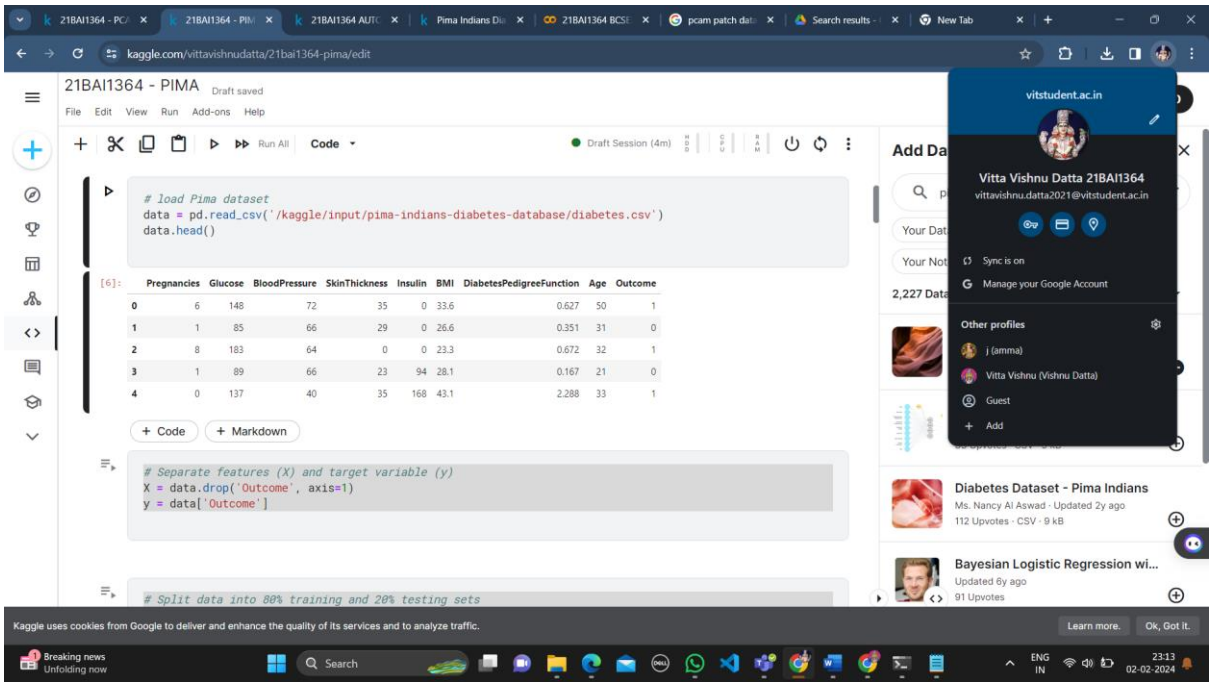


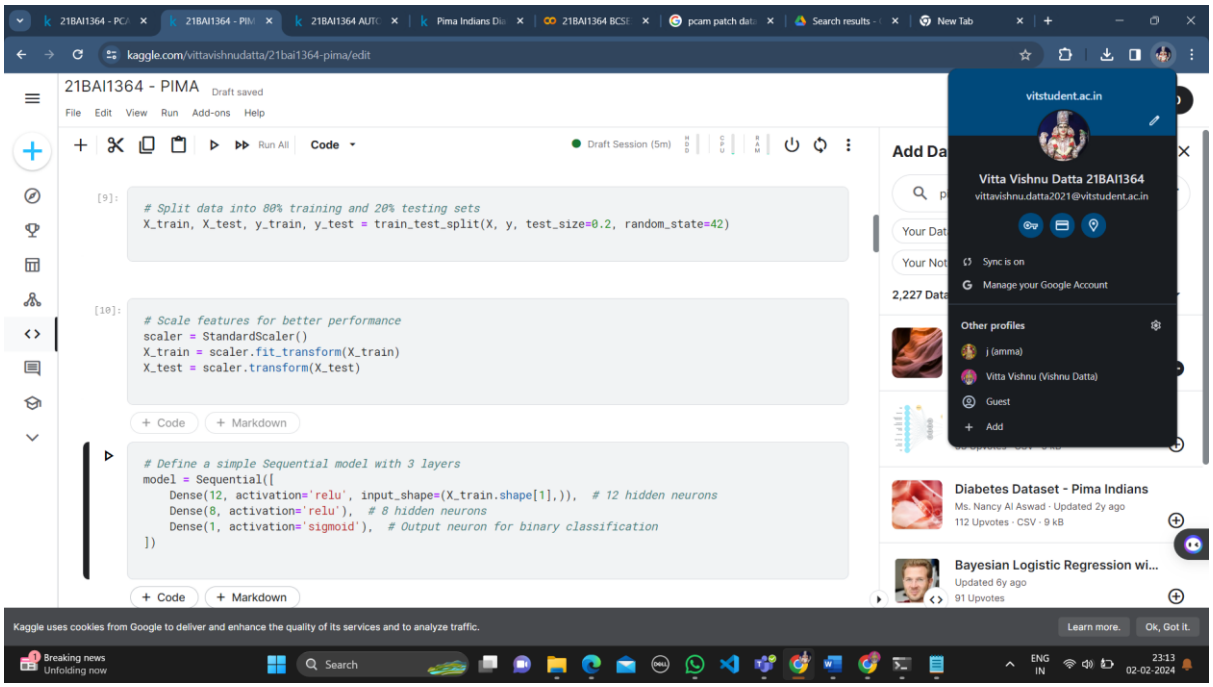
VIT®

Vellore Institute of Technology

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

CHENNAI



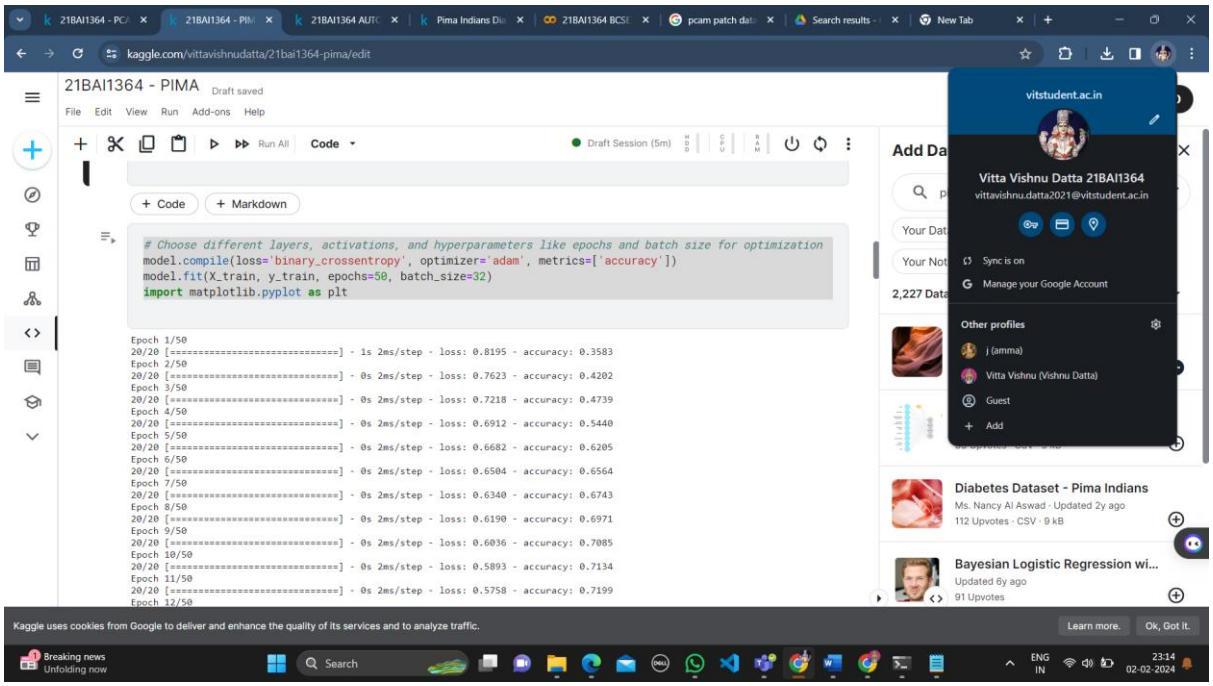


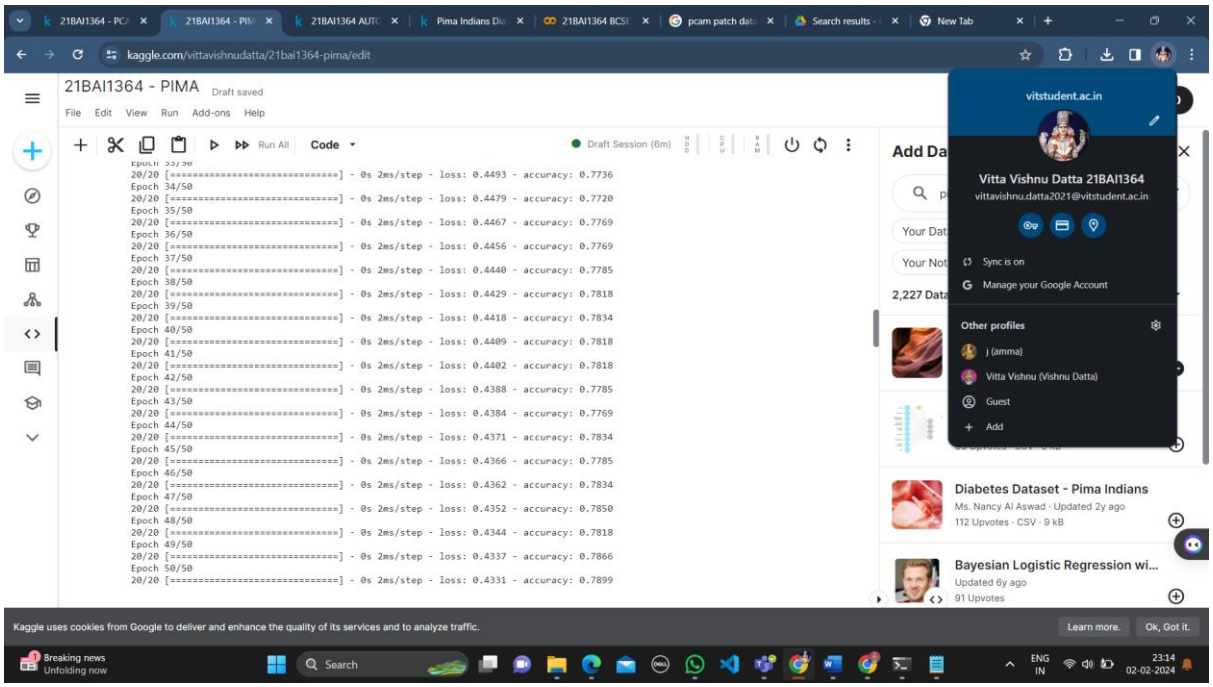
VIT®

Vellore Institute of Technology

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

CHENNAI



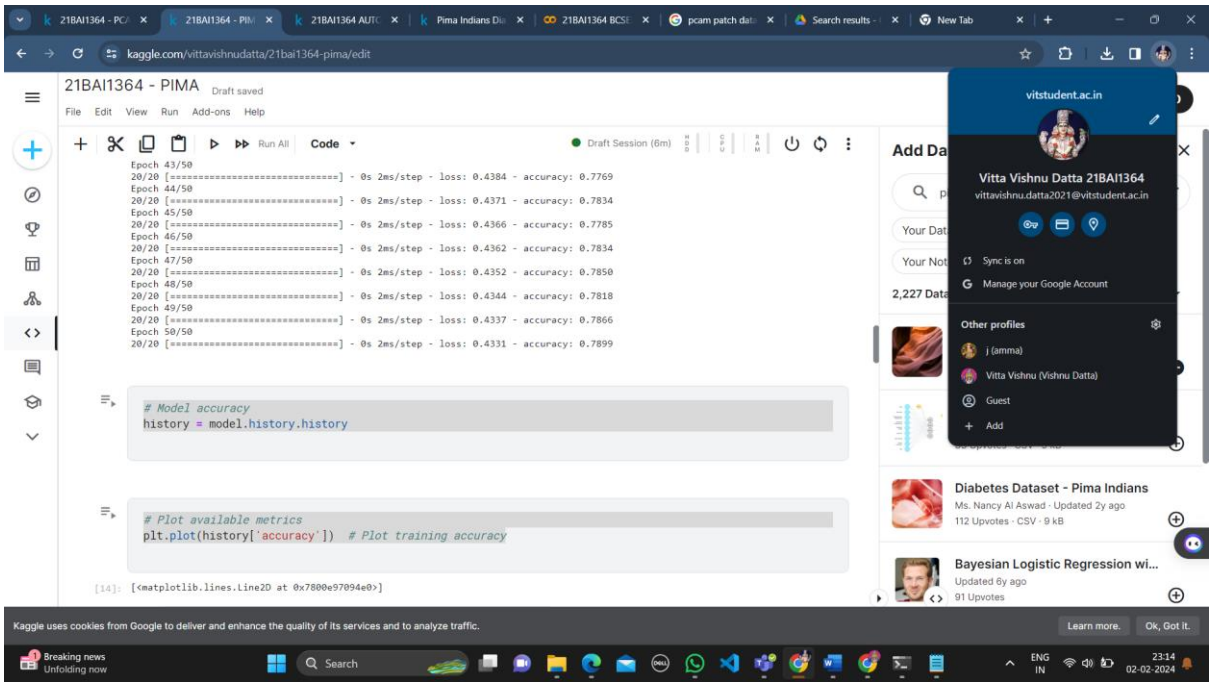


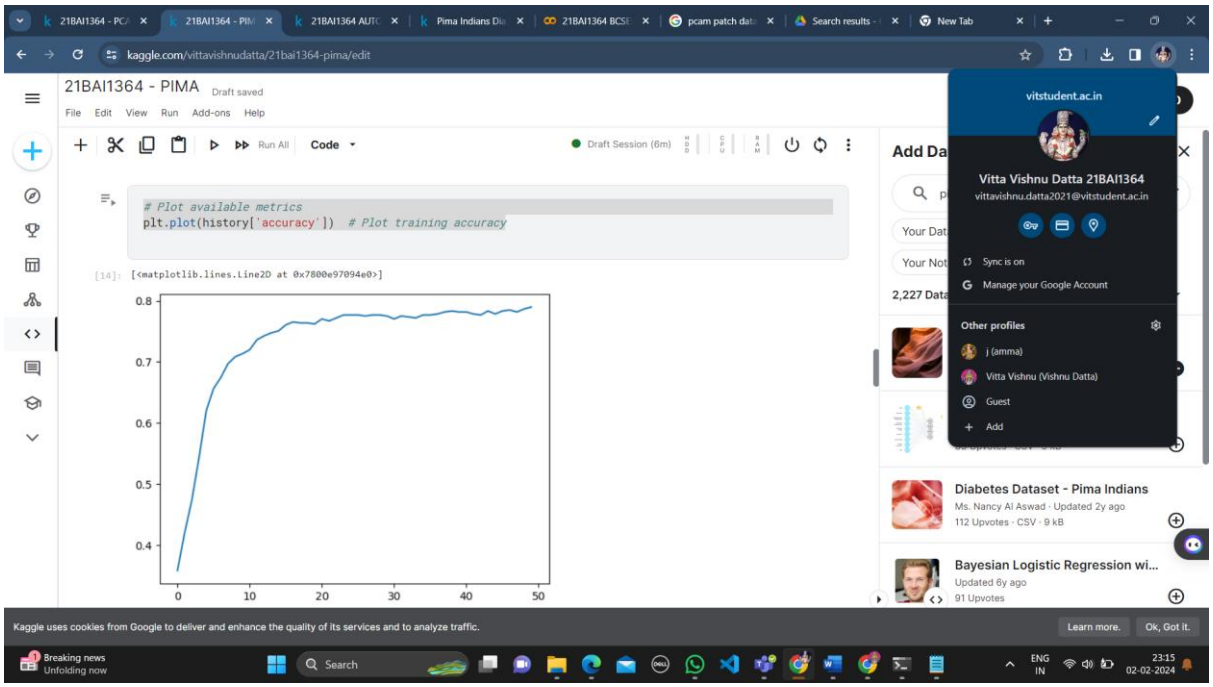
VIT®

Vellore Institute of Technology

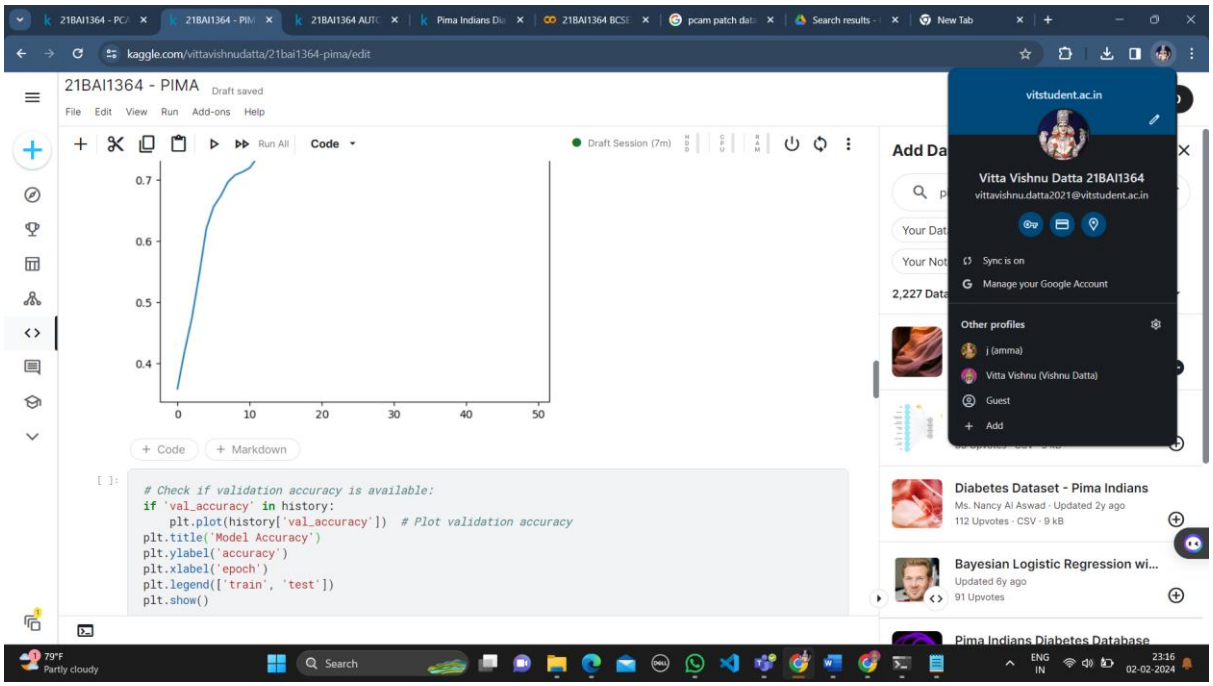
(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI





VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI



AUTO-TUNING OF PIMA DATASET:

!pip3 install --upgrade tensorflow

!pip3 installscikeras[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
```

```
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' ))
```

```
    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:

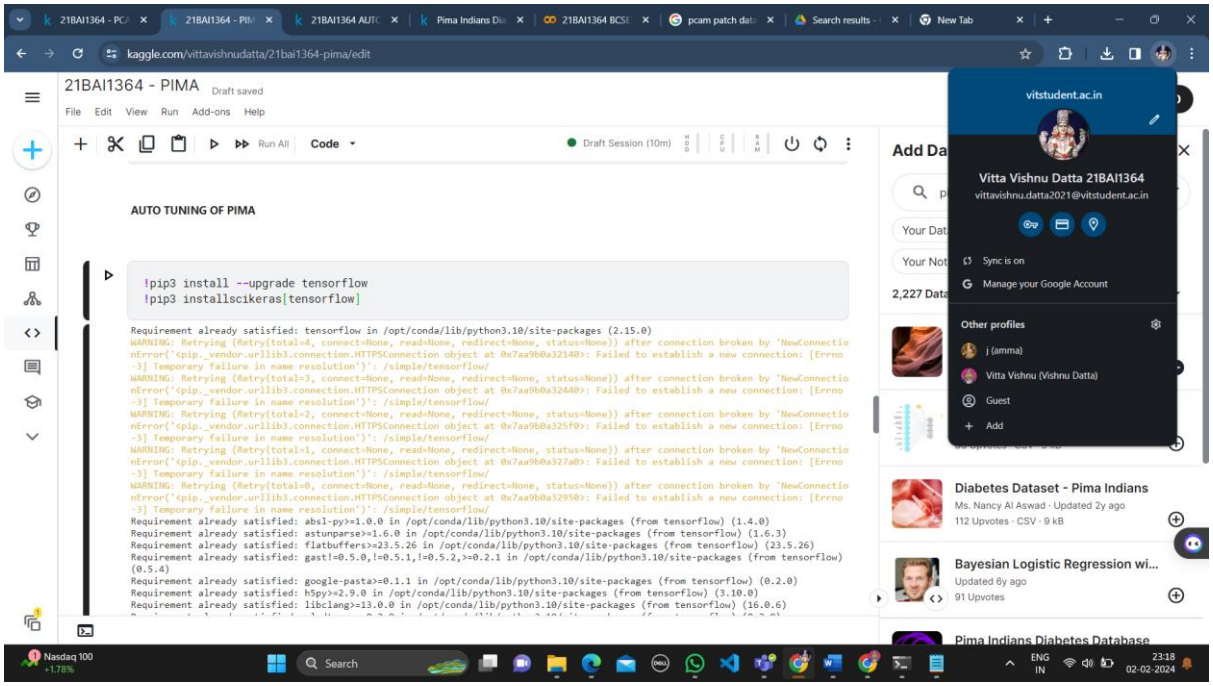
```
import numpy as np

from sklearn import datasets

from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score
```

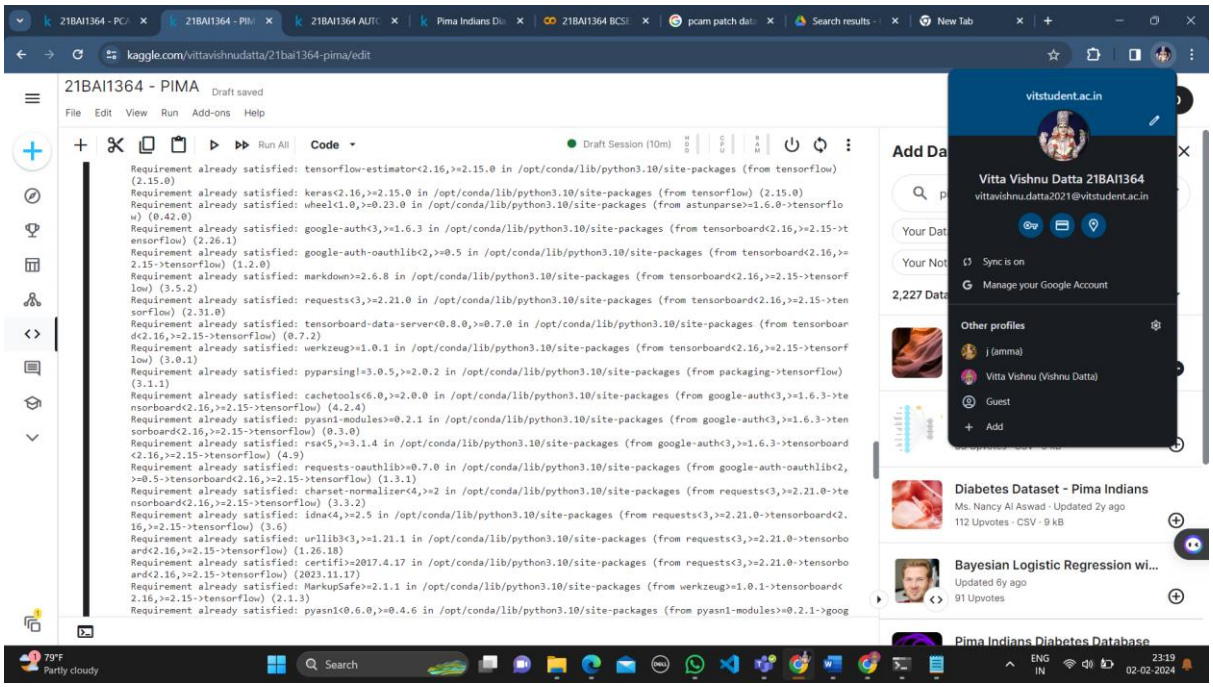
```
# Load the Pima Indians Diabetes dataset
pima_data = datasets.load_diabetes()
X = pima_data.data  # Features
y = pima_data.target  # Labels

# 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')
```

```
# 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:



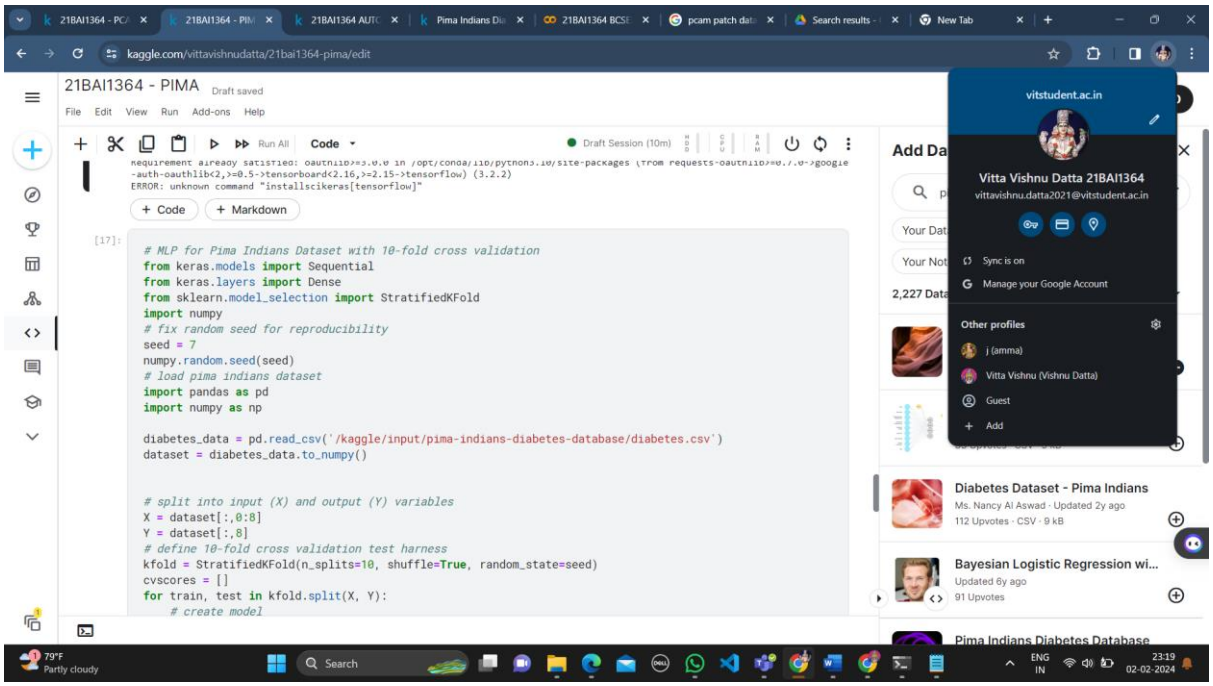


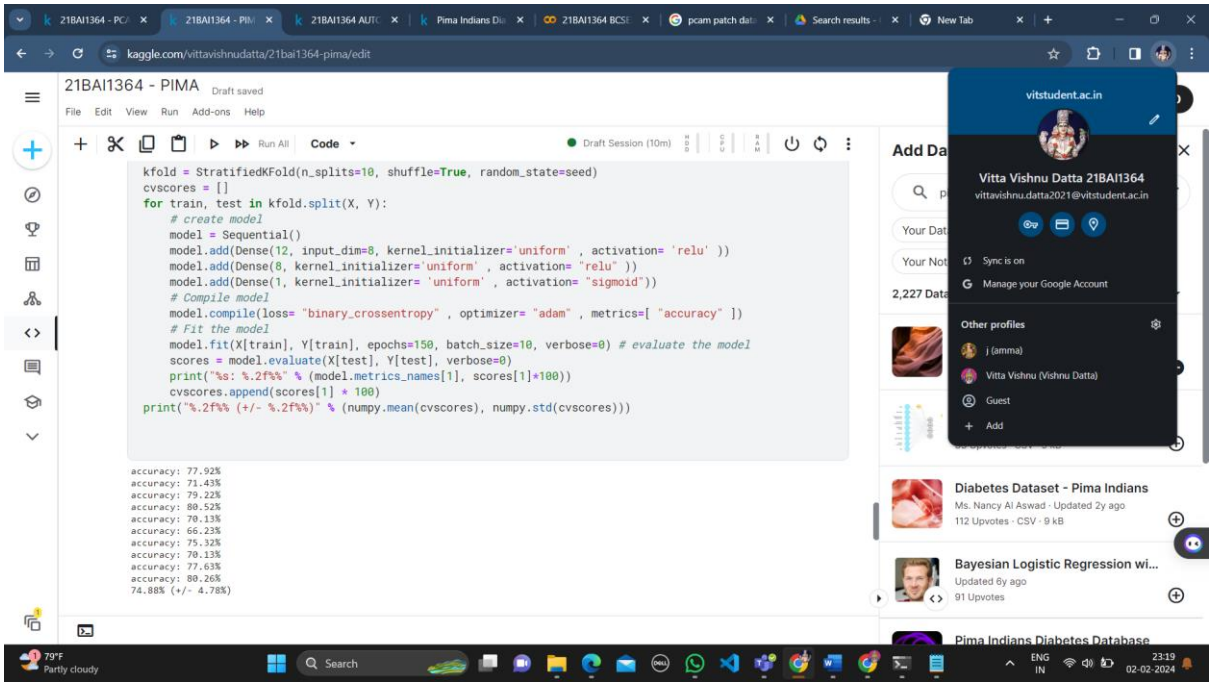
VIT®

Vellore Institute of Technology

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

CHENNAI



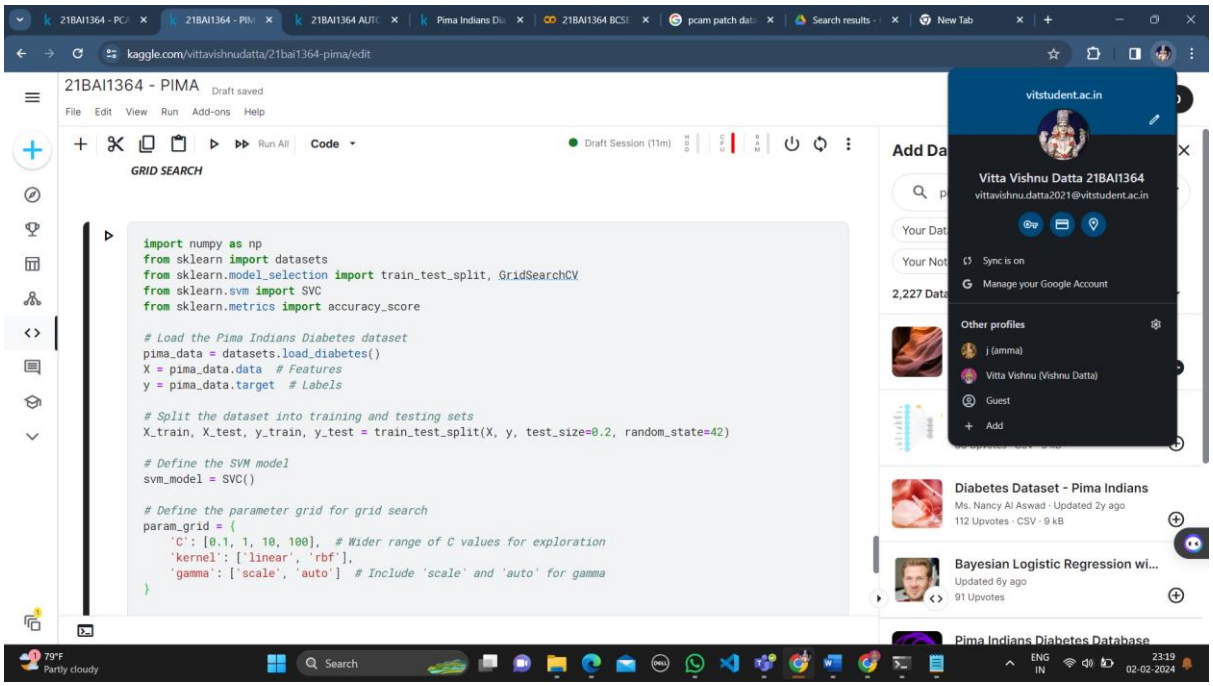


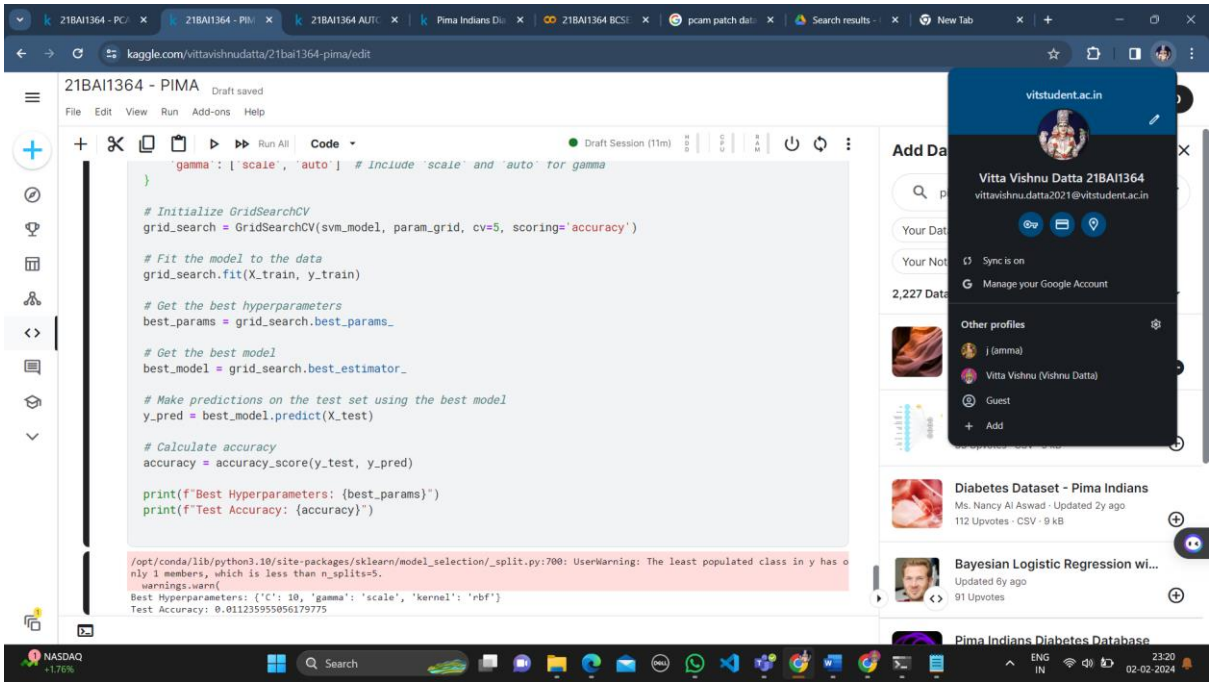
VIT®

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()
```

```
    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)
```

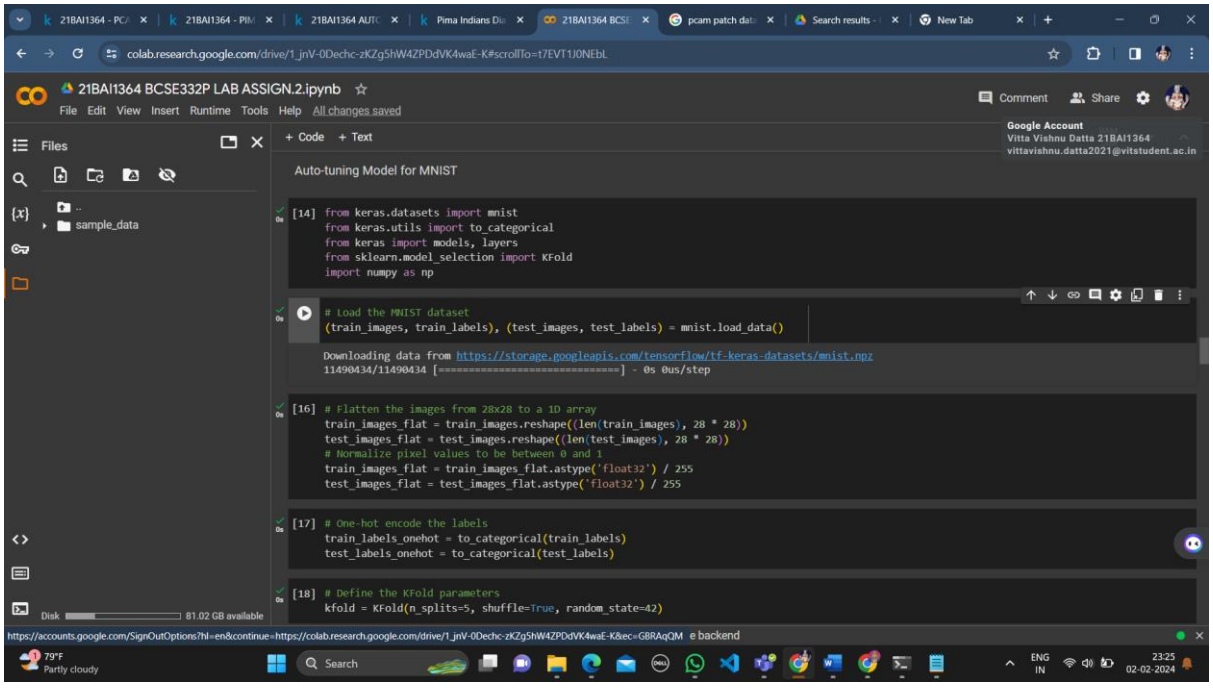
```
# Calculate and print the mean accuracy and standard deviation across
all folds
print("Mean Accuracy: %.2f%% (+/- %.2f%%)" % (np.mean(cvscores),
np.std(cvscores)))
# Define the model creation function
def create_model(optimizer='rmsprop', init='glorot_uniform'):
    model = Sequential()
    model.add(Dense(512, 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)
```

```
model1
# Define the hyperparameter grid

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

SCREENSHOTS OF THE CODE SNIPPET:

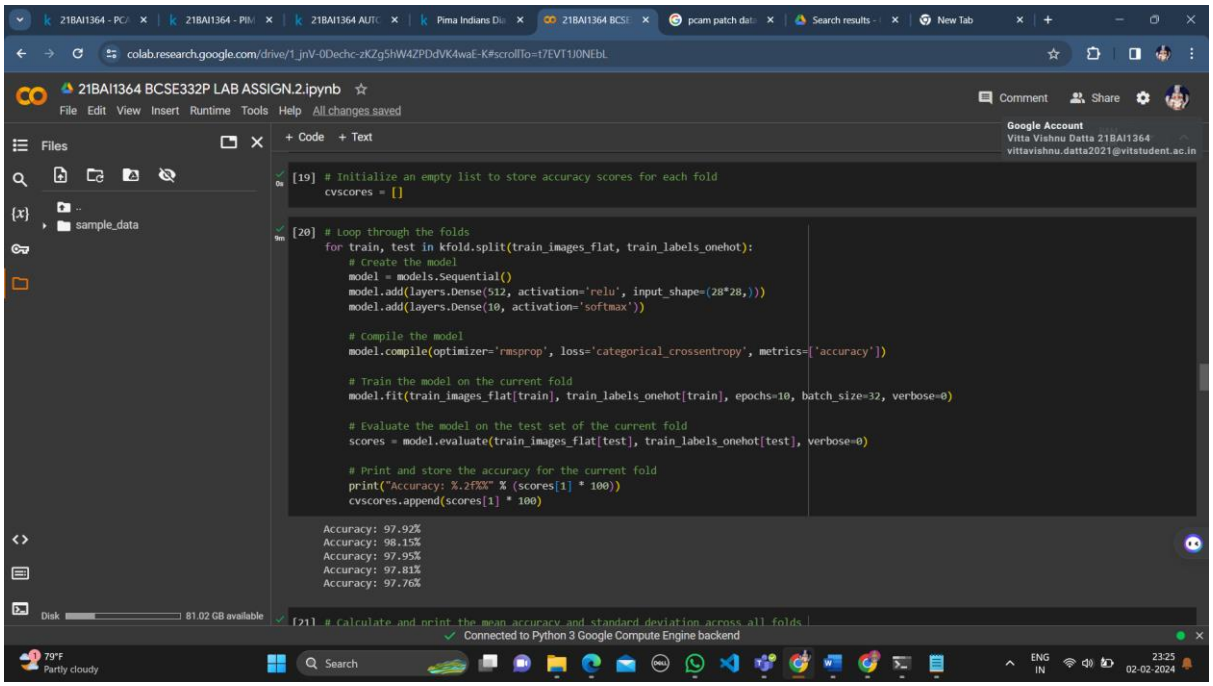


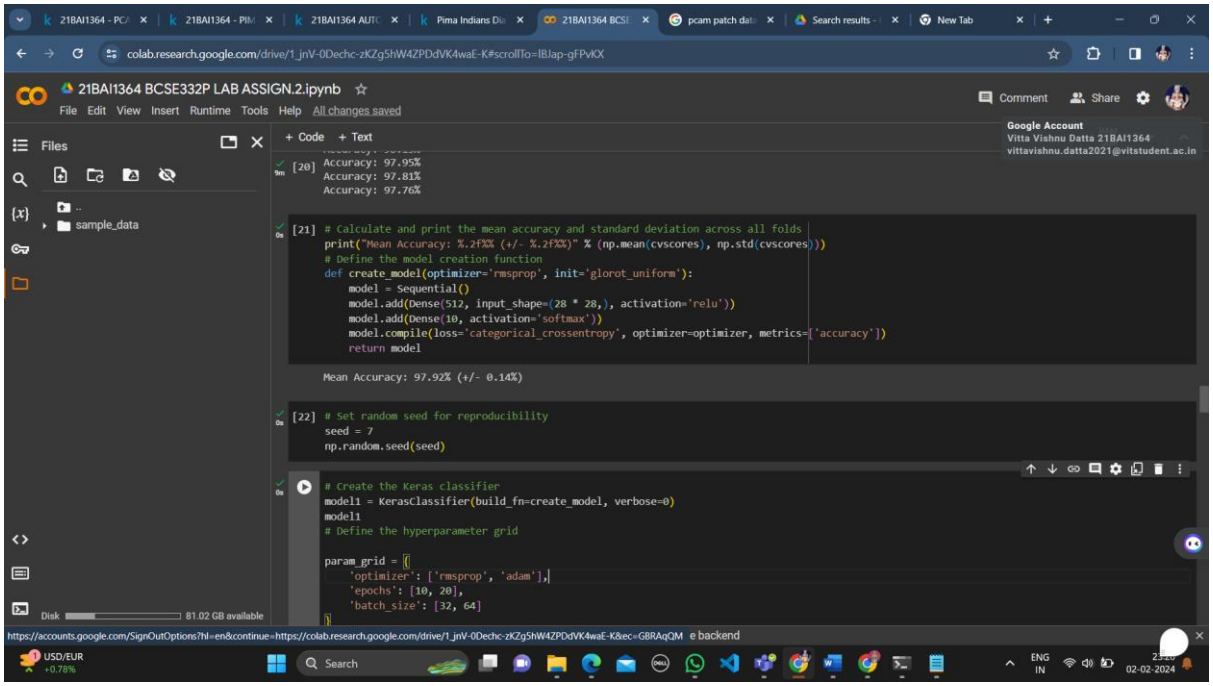
VIT®

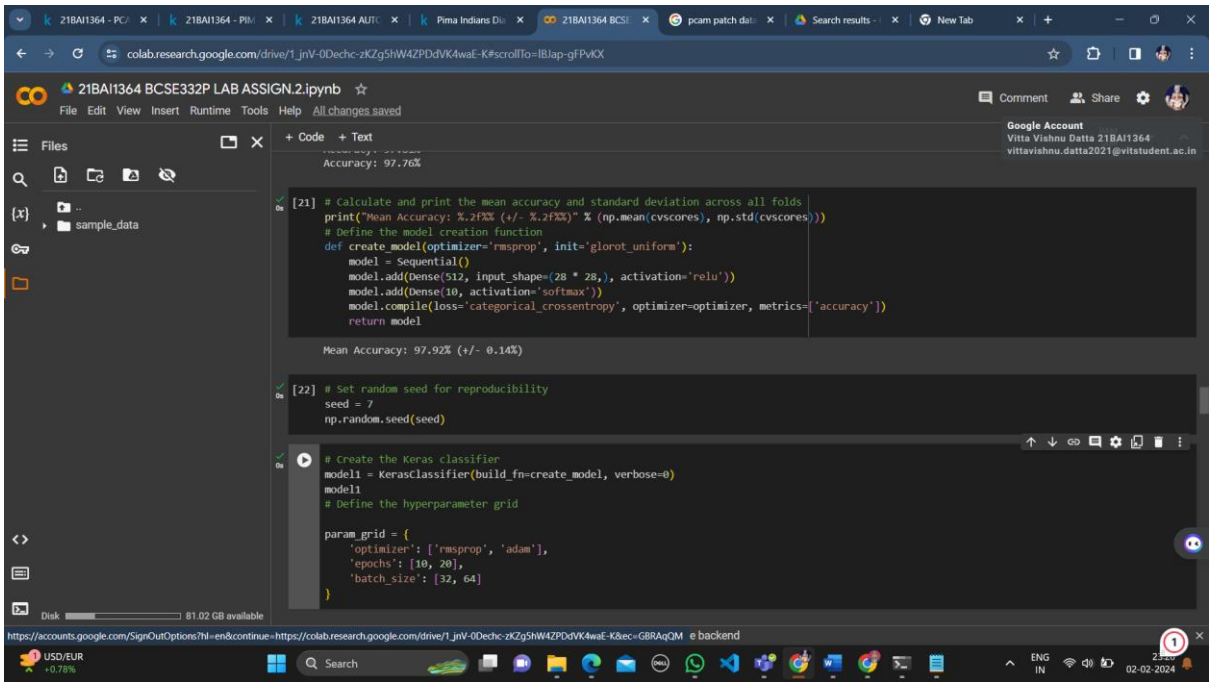
Vellore Institute of Technology

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

CHENNAI







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)
```

```
# 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
```

```
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
```

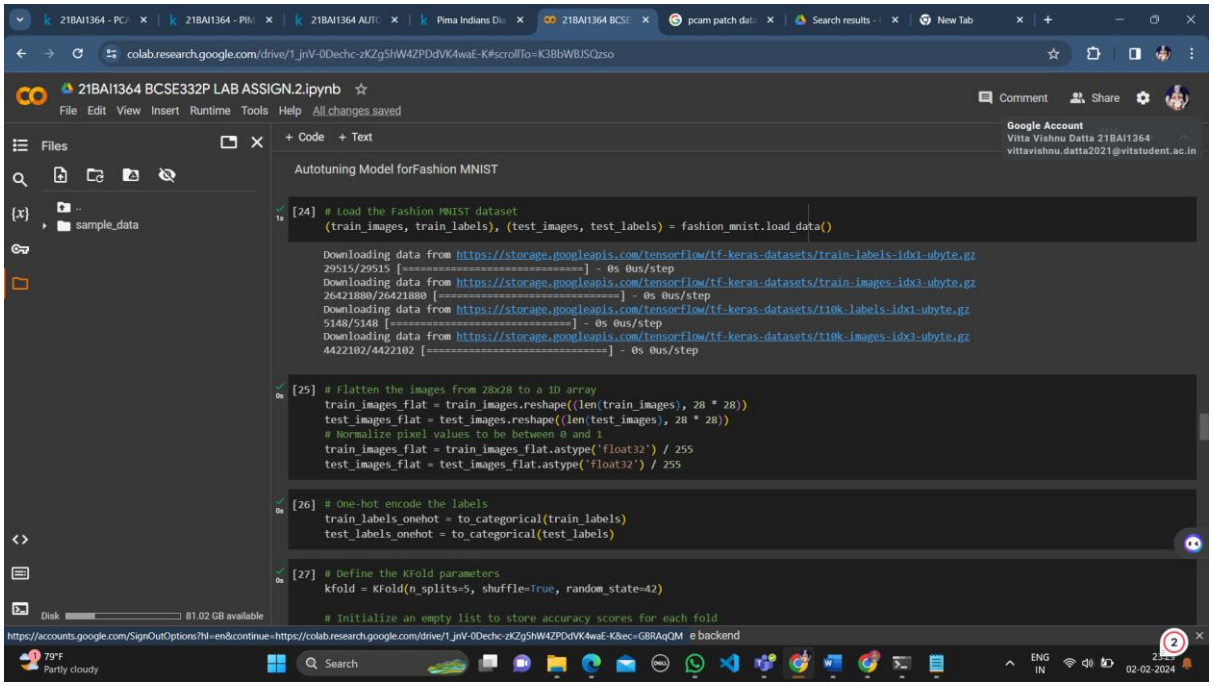
```
# Set random seed for reproducibility
seed = 7
np.random.seed(seed)
```

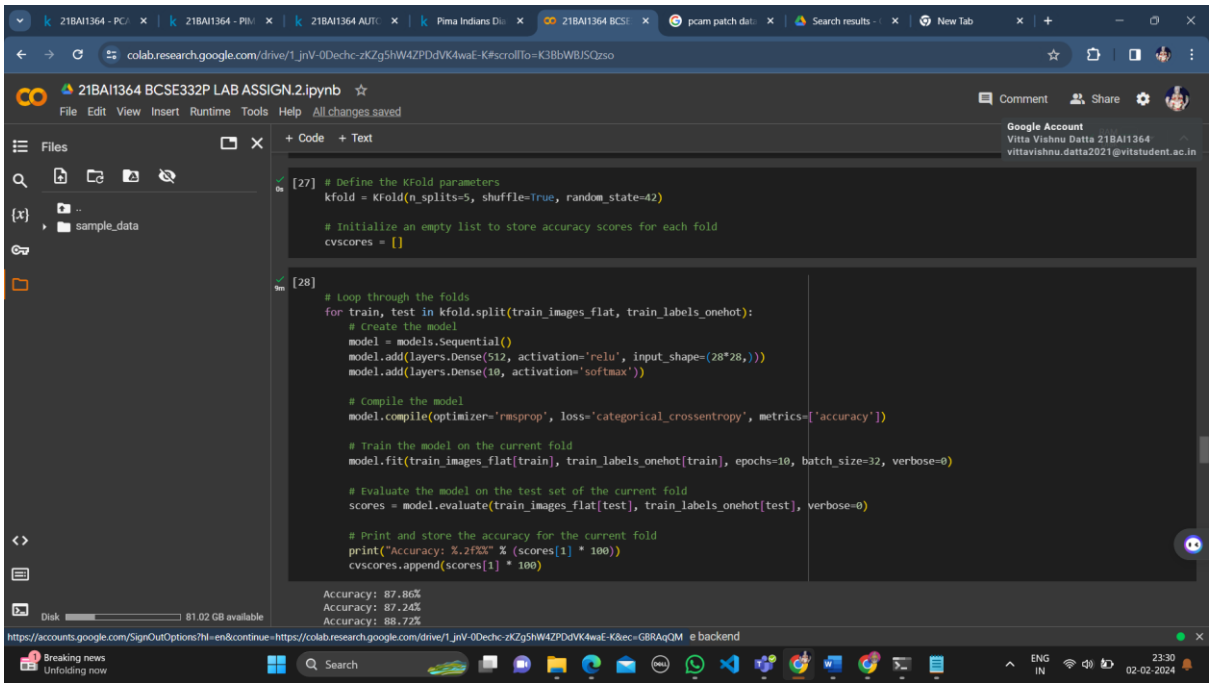
```
# Create the Keras classifier
model2 = KerasClassifier(build_fn=create_model, verbose=0)
model2
```

```
# Define the hyperparameter grid

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

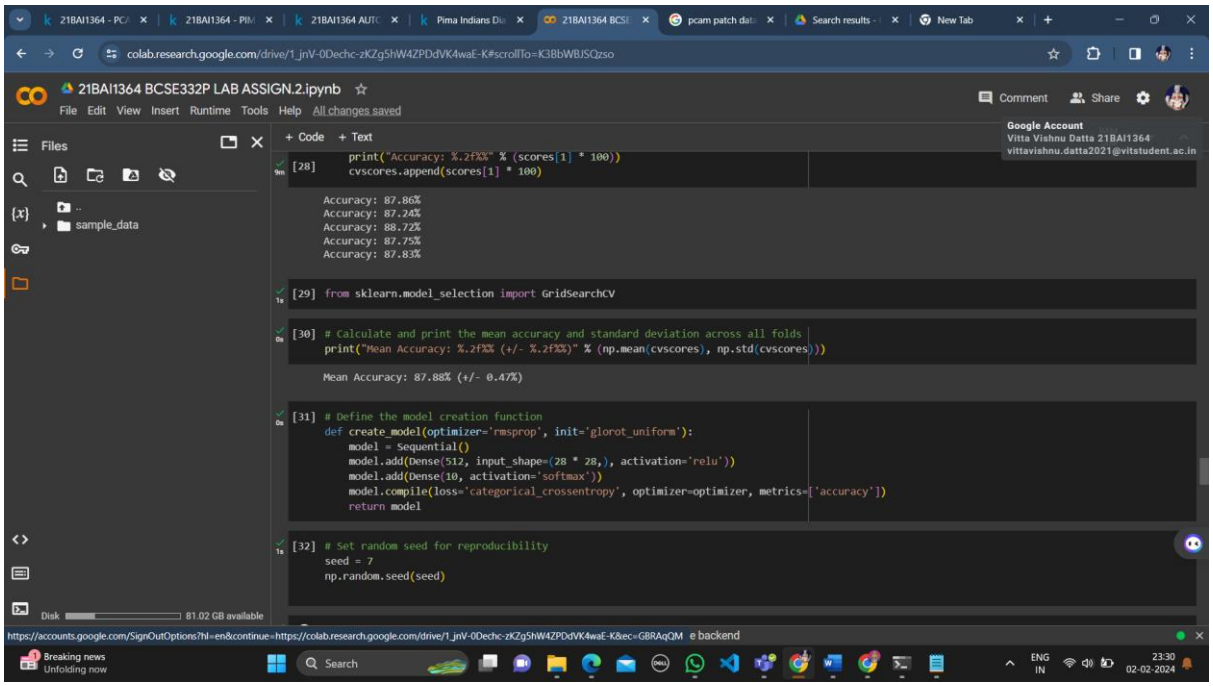
```
import h5py
import numpy as np
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

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
```

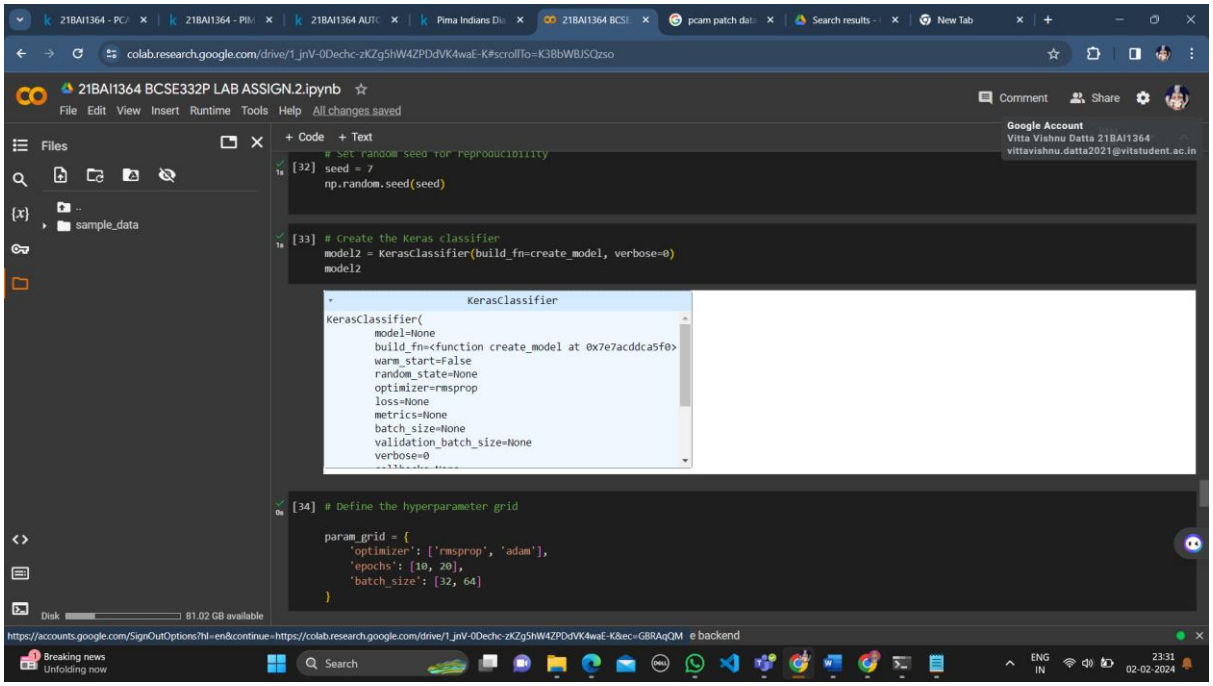
SCREENSHOTS OF THE CODE SNIPPET:





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 = []
```

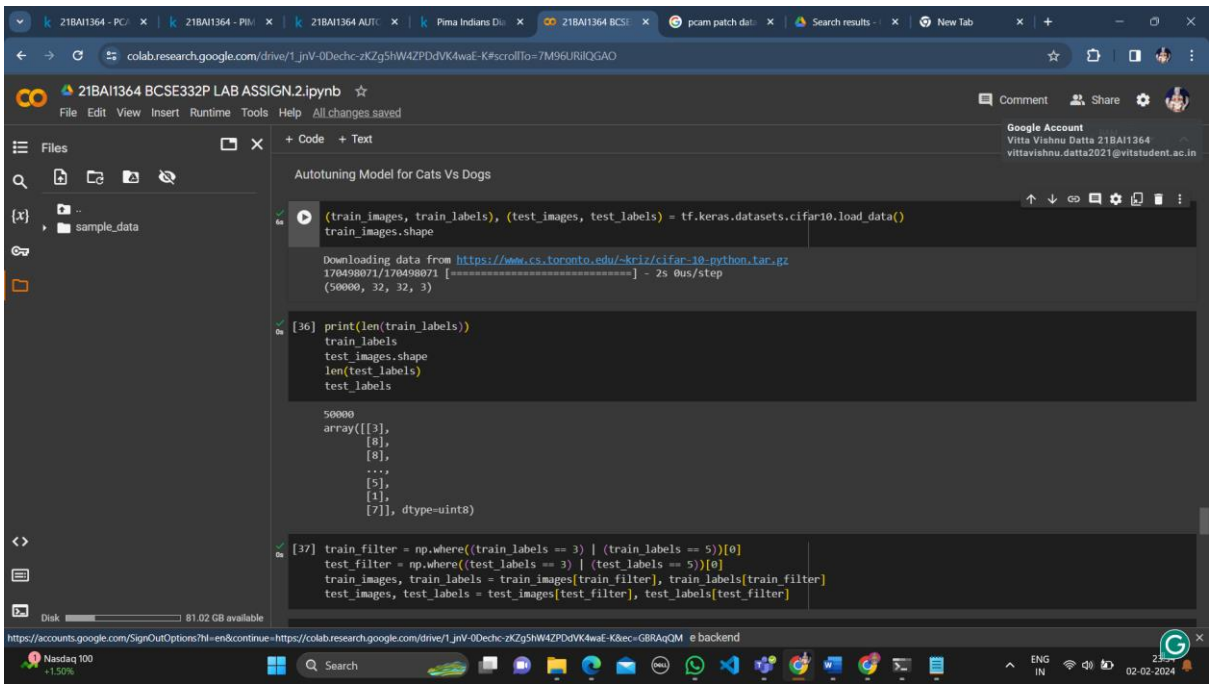
```
# Loop through the folds
for train, test in kfold.split(train_images, train_labels):
    # Create the model
    dogs_v_cats_model = models.Sequential()
```

```
    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'))
```

```
    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)
```

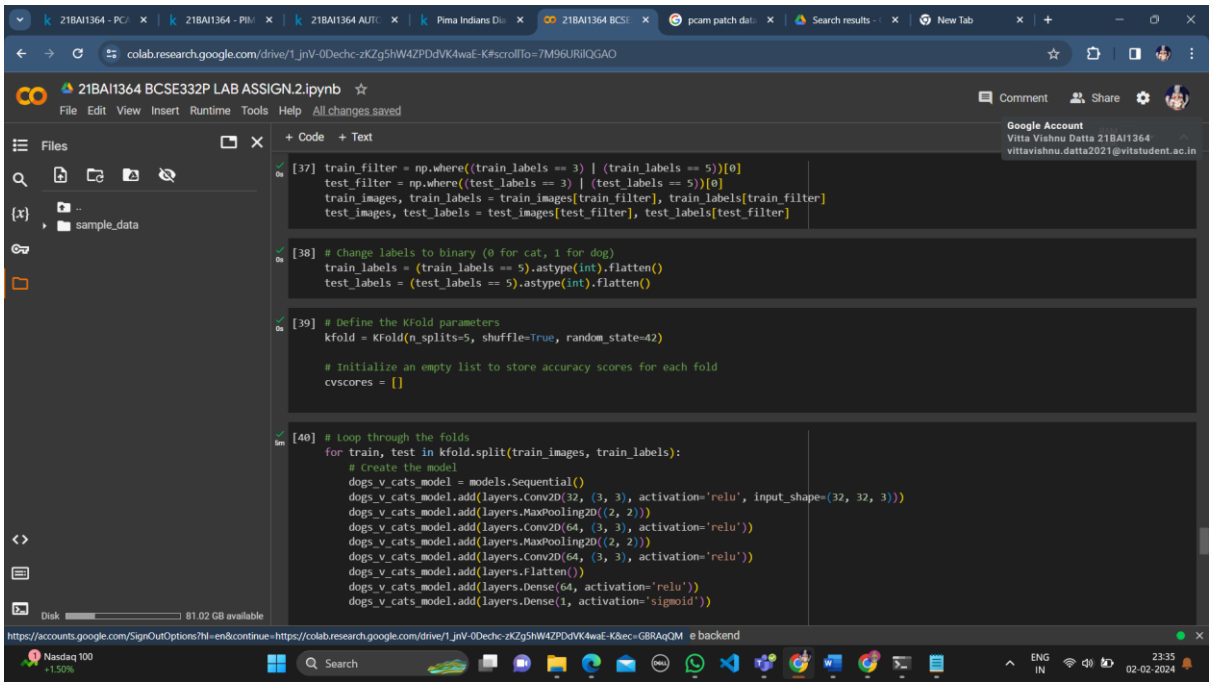
SCREENSHOTS OF THE CODE SNIPPET:

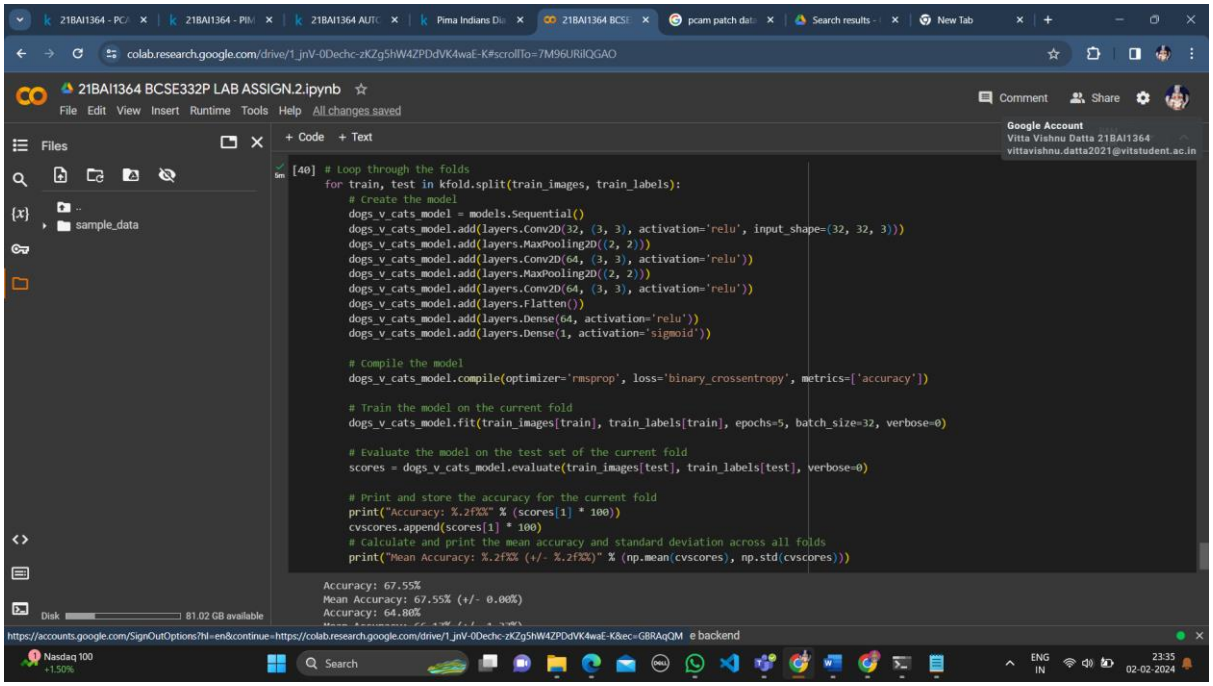


Vellore Institute of Technology

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

CHENNAI

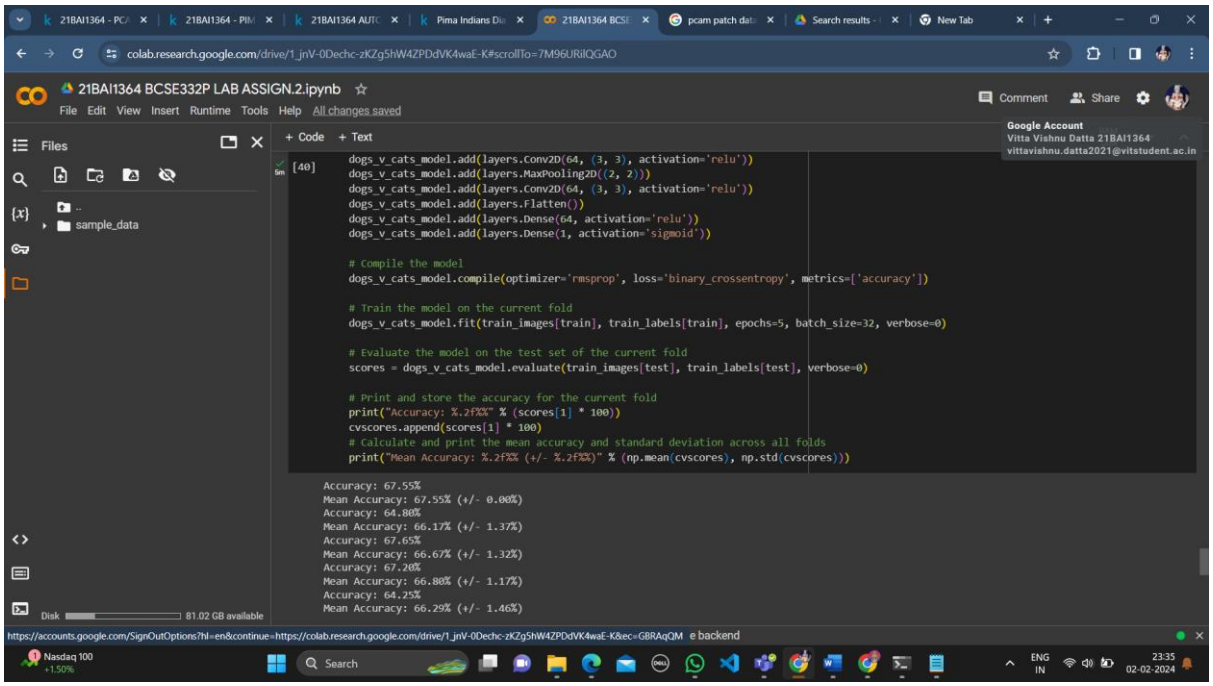


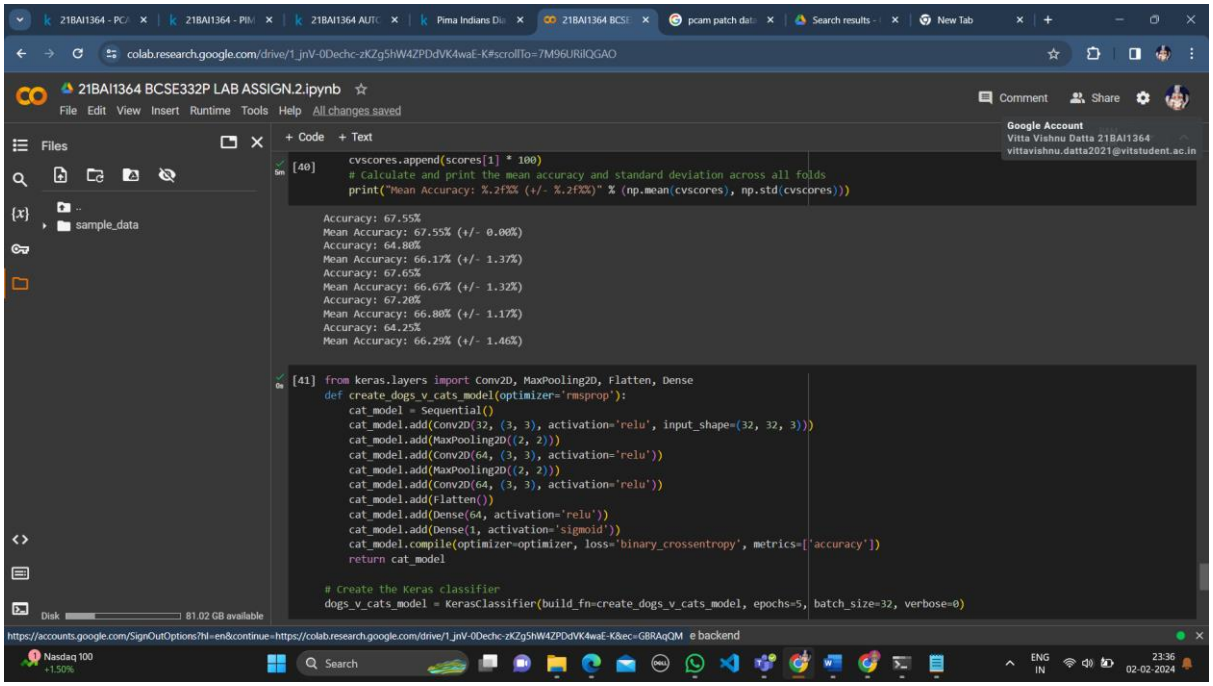


Vellore Institute of Technology

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

CHENNAI





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.