**Interim Report**

**Name: Vishwas B**

**Reg No: 2022MMA07ASB047**

**Supervisor’s Name: Dr. Sarit Maitra**

Contents

[**1. Latent Non-Gaussian Models** 2](#_Toc162614262)

[**2.Non-Gaussian statistical models** 3](#_Toc162614263)

[**3. ConvLSTM** 5](#_Toc162614264)

[**4. Support Vector Machines** 6](#_Toc162614265)

[**5. identify the data** 8](#_Toc162614266)

# **1. Latent Non-Gaussian Models**

import pandas as pd

from sklearn.metrics import confusion\_matrix, recall\_score, precision\_score

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

# Load data from CSV file

def load\_data(file\_path):

data = pd.read\_csv(file\_path)

return data

# Train a model and predict

def train\_and\_predict(data, target\_column):

X = data.drop(target\_column, axis=1)

y = data[target\_column]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = RandomForestClassifier()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

return y\_test, y\_pred

# Calculate confusion matrix, recall and precision

def calculate\_metrics(y\_test, y\_pred):

cm = confusion\_matrix(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred, average='macro')

precision = precision\_score(y\_test, y\_pred, average='macro')

return cm, recall, precision

# Main function

def main():

file\_path = 'your\_file\_path.csv' # replace with your file path

target\_column = 'target' # replace with your target column

data = load\_data(file\_path)

y\_test, y\_pred = train\_and\_predict(data, target\_column)

cm, recall, precision = calculate\_metrics(y\_test, y\_pred)

print('Confusion Matrix:', cm)

print('Recall:', recall)

print('Precision:', precision)

if \_\_name\_\_ == "\_\_main\_\_":

main()

# **2.Non-Gaussian statistical models**

import pandas as pd

from sklearn.metrics import confusion\_matrix, recall\_score, precision\_score

from sklearn.model\_selection import train\_test\_split

from sklearn.mixture import GaussianMixture

# Load data from CSV file

def load\_data(file\_path):

data = pd.read\_csv(file\_path)

return data

# Train a model and predict

def train\_and\_predict(data, target\_column):

X = data.drop(target\_column, axis=1)

y = data[target\_column]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = GaussianMixture(n\_components=2)

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

return y\_test, y\_pred

# Calculate confusion matrix, recall and precision

def calculate\_metrics(y\_test, y\_pred):

cm = confusion\_matrix(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred, average='macro')

precision = precision\_score(y\_test, y\_pred, average='macro')

return cm, recall, precision

# Main function

def main():

file\_path = 'your\_file\_path.csv' # replace with your file path

target\_column = 'target' # replace with your target column

data = load\_data(file\_path)

y\_test, y\_pred = train\_and\_predict(data, target\_column)

cm, recall, precision = calculate\_metrics(y\_test, y\_pred)

print('Confusion Matrix:', cm)

print('Recall:', recall)

print('Precision:', precision)

if \_\_name\_\_ == "\_\_main\_\_":

main()

# **3. ConvLSTM**

import pandas as pd

import numpy as np

from sklearn.metrics import confusion\_matrix, recall\_score, precision\_score

from sklearn.model\_selection import train\_test\_split

from keras.models import Sequential

from keras.layers import ConvLSTM2D, Dense, Flatten

from keras.utils import to\_categorical

# Load data from CSV file

def load\_data(file\_path):

data = pd.read\_csv(file\_path)

return data

# Reshape data for ConvLSTM

def reshape\_data(X, y, time\_steps=1):

X = np.array(X)

y = np.array(y)

X = X.reshape((X.shape[0], time\_steps, 1, X.shape[1], 1))

return X, y

# Train a model and predict

def train\_and\_predict(data, target\_column):

X = data.drop(target\_column, axis=1)

y = data[target\_column]

y = to\_categorical(y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

X\_train, y\_train = reshape\_data(X\_train, y\_train)

X\_test, y\_test = reshape\_data(X\_test, y\_test)

model = Sequential()

model.add(ConvLSTM2D(filters=64, kernel\_size=(1,3), activation='relu', input\_shape=(1, 1, X\_train.shape[2], 1)))

model.add(Flatten())

model.add(Dense(y\_train.shape[1], activation='softmax'))

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

model.fit(X\_train, y\_train, epochs=10, verbose=0)

y\_pred = model.predict\_classes(X\_test)

return y\_test.argmax(axis=1), y\_pred

# Calculate confusion matrix, recall and precision

def calculate\_metrics(y\_test, y\_pred):

cm = confusion\_matrix(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred, average='macro')

precision = precision\_score(y\_test, y\_pred, average='macro')

return cm, recall, precision

# Main function

def main():

file\_path = 'your\_file\_path.csv' # replace with your file path

target\_column = 'target' # replace with your target column

data = load\_data(file\_path)

y\_test, y\_pred = train\_and\_predict(data, target\_column)

cm, recall, precision = calculate\_metrics(y\_test, y\_pred)

print('Confusion Matrix:', cm)

print('Recall:', recall)

print('Precision:', precision)

if \_\_name\_\_ == "\_\_main\_\_":

main()

# **4. Support Vector Machines**

import pandas as pd

from sklearn.metrics import confusion\_matrix, recall\_score, precision\_score

from sklearn.model\_selection import train\_test\_split

from sklearn import svm

import matplotlib.pyplot as plt

from sklearn.metrics import plot\_confusion\_matrix

# Load data from CSV file

def load\_data(file\_path):

data = pd.read\_csv(file\_path)

return data

# Train a model and predict

def train\_and\_predict(data, target\_column):

X = data.drop(target\_column, axis=1)

y = data[target\_column]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = svm.SVC()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

return model, X\_test, y\_test, y\_pred

# Calculate confusion matrix, recall and precision

def calculate\_metrics(y\_test, y\_pred):

cm = confusion\_matrix(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred, average='macro')

precision = precision\_score(y\_test, y\_pred, average='macro')

return cm, recall, precision

# Plot confusion matrix

def plot\_cm(model, X\_test, y\_test):

plot\_confusion\_matrix(model, X\_test, y\_test)

plt.show()

# Main function

def main():

file\_path = 'your\_file\_path.csv' # replace with your file path

target\_column = 'target' # replace with your target column

data = load\_data(file\_path)

model, X\_test, y\_test, y\_pred = train\_and\_predict(data, target\_column)

cm, recall, precision = calculate\_metrics(y\_test, y\_pred)

print('Confusion Matrix:', cm)

print('Recall:', recall)

print('Precision:', precision)

plot\_cm(model, X\_test, y\_test)

if \_\_name\_\_ == "\_\_main\_\_":

main()

# **5. identify the data**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.neighbors import KernelDensity

# Load data from CSV (replace 'your\_data.csv' with your actual file)

data = pd.read\_csv('your\_data.csv')

# Visualize the data (e.g., plot histograms)

plt.hist(data['column\_name'], bins=30, density=True, alpha=0.6, color='b')

# Fit a KDE to the data

kde = KernelDensity(bandwidth=0.1) # Adjust bandwidth as needed

kde.fit(data[['column\_name']])

# Generate new samples from the KDE

num\_samples = 1000

new\_samples = kde.sample(num\_samples)

# Plot the KDE estimate

x\_vals = np.linspace(data['column\_name'].min(), data['column\_name'].max(), 1000)

log\_dens = kde.score\_samples(x\_vals.reshape(-1, 1))

plt.plot(x\_vals, np.exp(log\_dens), 'r-', label='KDE')

plt.xlabel('Your Feature')

plt.ylabel('Density')

plt.title('Non-Gaussian Data Distribution')

plt.legend()

plt.show()

# Now 'new\_samples' contains non-Gaussian data points