A Project Report submitted

For Simulation & Modelling (UCS751)

by

Anirudh Anand	**101903004**
Yuvraj Gupta	**101903549**

Submitted to

Mr. Sanjeev Rao

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY,

(A DEEMED TO BE UNIVERSITY),

PATIALA, PUNJAB

INDIA

Nov 2022

Traffic Flow Prediction

Objective

Traffic flow prediction is an essential part of the intelligent transport system. This is the accurate estimation of traffic flow in a given region at a particular interval of time in the future. The study of traffic forecasting is useful in mitigating congestion and make safer and cost-efficient travel.

While traditional models use shallow networks, there has been an exponential growth in the number of vehicles in recent times and these traditional machine learning models fail to work in current scenarios. In our paper, we review some of the latest works in deep learning for traffic flow prediction.

Many deep learning architectures include Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Restricted Boltzmann Machines (RBM), and Stacked Auto Encoder (SAE). These deep learning models use multiple layers to extract higher level of features from raw input progressively. The latest deep learning models developed to tackle this very problem are reviewed and due to the complexity of transport networks, this review gives the reader information about how various factors influence these models and what models work best in different scenarios.

Machine Learning technology has developed so much that we can now help the community by detecting breast cancer on the basis of various features such as radius, texture, compactness etc. We have used the K-nearest neighbor (KNN) algorithm to develop a data model which predicts whether the tumor is benign or malignant.

The Traffic flow dataset by Coplin has been used for the training and testing of the data model created by us.

Dataset

The dataset used in this project is the Coplin Traffic flow Dataset. The dataset has 30 parameters defined on the basis of which we determine whether the result is malignant or benign.

The different parameters are states as follows:

radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave points_se
symmetry_se
fractal_dimension_se
radius_worst
texture_worst
perimeter_worst
area_worst
smoothness_worst
compactness_worst
concavity_worst
concave points_worst
symmetry_worst
fractal_dimension_worst

Methodology

The Data Model in this project is made using the K-nearest neighbor algorithm. Among the supervised machine learning algorithms, K-nearest neighbors (KNN) is one of the most effective techniques. It performs classification on certain data points. The KNN algorithm is a type of supervised ML algorithm that can be used for both classifications as well as regression predictive problems. It uses 'attribute similarity' to predict the values of new data-points and then the new data point will be assigned a value based on how closely it matches the points in the training set.

Algorithm

Step-1: **Select the number K of the neighbors**

Step-2: **Calculate the Euclidean distance of** K number of neighbors

Step-3: **Take the K nearest neighbors as per the calculated Euclidean distance.**

Step-4: **Among these k neighbors, count the number of the data points in each category.**

Step-5: **Assign the new data points to that category for which the number of the neighbor is maximum.**

Step-6: **Our model is ready.**

The K value for our dataset is 23, because the optimal K values is the square root of the number of entries in the dataset, i.e.,569, which comes out to be 23.85372 Rounding it down to 23 is better because we want an odd K value as an odd number so that we can calculate a clear majority in the case where only two groups are possible.

Code

traffice_flow_prediction.py

C: > Users > pnb > Downloads > traffice_flow_prediction.py

```python
# -*- coding: utf-8 -*-
"""traffice_flow_prediction.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1pKuw7QWql39bdV88hWsu29zExOMBeI72
"""

!unzip archive.zip


import pandas as pd
import glob

path = '/content/DataSet'
all_files = glob.glob(path + "/*.csv")

li = []

for filename in all_files:
    df = pd.read_csv(filename, index_col=None, header=0)
    li.append(df)

df = pd.concat(li, axis=0, ignore_index=True)
print(df.head(3))

"""### Descriptive Analysis"""

df.describe()

```

```
df.describe()

"""### Getting new features"""

df['timestamp'] = df['timestamp'].astype('datetime64[ns]')
df['weekday'] = df['timestamp'].dt.weekday

# Feature engineering with the date
df['year']= df['timestamp'].dt.year
df['month']= df['timestamp'].dt.month
df['day']= df['timestamp'].dt.day

df.head(3)

original_df = df.copy()

import matplotlib.pyplot as plt
plt.style.use('ggplot')

plt.figure(figsize=(50, 8))
mean_group = df[['timestamp','hourly_traffic_count']].groupby(['timestamp'])['hourly_traffic_count'].mean()
plt.plot(mean_group)
plt.title('Time Series - Average')
plt.show()

plt.figure(figsize=(50, 8))
median_group = df[['timestamp','hourly_traffic_count']].groupby(['timestamp'])['hourly_traffic_count'].median()
plt.plot(median_group, color = 'b')
plt.title('Time Series - median')
plt.show()
```

```
plt.show()

df['weekday_num'] = df['weekday']
df['weekday'].replace(0,'01 - Monday',inplace=True)
df['weekday'].replace(1,'02 - Tuesday',inplace=True)
df['weekday'].replace(2,'03 - Wednesday',inplace=True)
df['weekday'].replace(3,'04 - Thursday',inplace=True)
df['weekday'].replace(4,'05 - Friday',inplace=True)
df['weekday'].replace(5,'06 - Saturday',inplace=True)
df['weekday'].replace(6,'07 - Sunday',inplace=True)

train_group = df.groupby(["month", "weekday"])['hourly_traffic_count'].mean().reset_index()
train_group = train_group.pivot('weekday','month','hourly_traffic_count')
train_group.sort_index(inplace=True)

import seaborn as sns
sns.set(font_scale=1.2)

f, ax = plt.subplots(figsize=(8, 8))
sns.heatmap(train_group, annot=False, ax=ax, fmt="d", linewidths=2)
plt.title('Web Traffic Months cross Weekdays')
plt.show()

"""##### We can clearly see that there is no activity during the December and people during Octobor contributed the highest activity """

times_series_means =  pd.DataFrame(mean_group).reset_index(drop=False)
times_series_means['weekday'] = times_series_means['timestamp'].apply(lambda x: x.weekday())
times_series_means['Date_str'] = times_series_means['timestamp'].apply(lambda x: str(x))
times_series_means[['year','month','day']] = pd.DataFrame(times_series_means['Date_str'].str.split('-',2).tolist(), columns = ['year','month','
date_staging = pd.DataFrame(times_series_means['day'].str.split(' ',2).tolist(), columns = ['day','other'])
times_series_means['day'] = date_staging['day']*1
```

```python
date_staging = pd.DataFrame(times_series_means['day'].str.split(' ',2).tolist(), columns = ['day','other'])
times_series_means['day'] = date_staging['day']*1
times_series_means.drop('Date_str',axis = 1, inplace =True)
del times_series_means['timestamp']
times_series_means.head()

"""#### Train/Test Preparation"""

from sklearn.model_selection import train_test_split

X, y = times_series_means.drop(['hourly_traffic_count','year'],axis=1), times_series_means['hourly_traffic_count']
trainx, testx, trainy, testy = train_test_split(X, y, test_size=0.2)


# Linear Model
from sklearn.ensemble import GradientBoostingRegressor, AdaBoostRegressor
from sklearn.metrics import mean_absolute_error, r2_score

def modelisation(x_tr, y_tr, x_ts, y_ts, model):
    # Modelisation with all product
    model.fit(x_tr, y_tr)

    prediction = model.predict(x_ts)
    r2 = r2_score(y_ts.to_numpy(), model.predict(x_ts))
    mae = mean_absolute_error(y_ts.to_numpy(), model.predict(x_ts))
    print ("-----------------------------------------------")
    print ("mae with 80% of the data to train:", mae)
    print ("-----------------------------------------------")

    return  prediction, model
```

C: > Users > pnb > Downloads > traffice_flow_prediction.py

```python
    return  prediction, model

model =  AdaBoostRegressor(n_estimators = 5000, random_state = 42, learning_rate=0.01)

prediction, clr = modelisation(trainx, trainy, testx, testy, model)

import numpy as np
plt.figure(figsize=(16, 4))
line_up, = plt.plot(prediction,label='Prediction', color="red")
plt.ylabel('Series')
plt.legend(handles=[line_up])
plt.title('Performance of predictions - Benchmark Predictions')
plt.show()

plt.figure(figsize=(16, 6))
line_down, = plt.plot(np.array(testy),label='Reality')
plt.ylabel('Series')
plt.legend(handles=[line_down])
plt.title('Performance of predictions - Benchmark Reality')
plt.show()

trainx.shape

"""### Keras LSTM"""

trainx.head()

from keras.models import Sequential
from keras.layers import Dense, LSTM
from pandas import DataFrame, concat
from sklearn.preprocessing import MinMaxScaler
```

C: > Users > pnb > Downloads > traffice_flow_prediction.py

```python
    return  prediction, model

model =  AdaBoostRegressor(n_estimators = 5000, random_state = 42, learning_rate=0.01)

prediction, clr = modelisation(trainx, trainy, testx, testy, model)

import numpy as np
plt.figure(figsize=(16, 4))
line_up, = plt.plot(prediction,label='Prediction', color="red")
plt.ylabel('Series')
plt.legend(handles=[line_up])
plt.title('Performance of predictions - Benchmark Predictions')
plt.show()

plt.figure(figsize=(16, 6))
line_down, = plt.plot(np.array(testy),label='Reality')
plt.ylabel('Series')
plt.legend(handles=[line_down])
plt.title('Performance of predictions - Benchmark Reality')
plt.show()

trainx.shape

"""### Keras LSTM"""

trainx.head()

from keras.models import Sequential
from keras.layers import Dense, LSTM
from pandas import DataFrame, concat
from sklearn.preprocessing import MinMaxScaler
```


```python
from keras.models import Sequential
from keras.layers import Dense, LSTM
from pandas import DataFrame, concat
from sklearn.preprocessing import MinMaxScaler
```


```python


X, y = times_series_means.drop(['hourly_traffic_count','year'],axis=1), times_series_means['hourly_traffic_count']
scaler = MinMaxScaler(feature_range=(0, 1))
scaled = scaler.fit_transform(X)

trainx, testx, trainy, testy = train_test_split(scaled, y, test_size=0.2)


trainx = trainx.reshape((trainx.shape[0], 1, trainx.shape[1]))
testx = testx.reshape((testx.shape[0], 1, testx.shape[1]))

# design network
model = Sequential()
model.add(LSTM(50, input_shape=(trainx.shape[1], trainx.shape[2])))
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam')
# fit network
history = model.fit(trainx, trainy, epochs=50, batch_size=8, validation_data=(testx, testy), verbose=2, shuffle=False)

"""#### Plot history and Evaluation"""

from sklearn.metrics import mean_squared_error

plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
```

```python

plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.legend()
plt.show()

# make a prediction
yhat = model.predict(testx)
# calculate RMSE
rmse = np.sqrt(mean_squared_error(testy, yhat))
print('Test RMSE: %.3f' % rmse)
```

Output

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Accurracy : 91.23711340206185%
PS C:\Users\a\Desktop\ML projects> & C:/Users
Accurracy : 94.53551912568307%
PS C:\Users\a\Desktop\ML projects> & C:/Users
Accurracy : 93.92265193370166%
PS C:\Users\a\Desktop\ML projects>
```

Conclusion

This project helped us understand the working of data models using KNN algorithm. Machine learning approaches have been increasing rapidly in the practical life due to their monumental performance in predicting and classifying problems.

The average accuracy of the data model is 93.23% with the maximum accuracy reached 94.5%.

References

The dataset has been taken from the following link:

https://www.kaggle.com/datasets/coplin/traffic

The data Model has been made with the help of the following videos:

https://www.youtube.com/watch?v=4HKqjENq9OU

https://www.youtube.com/watch?v=2btpg2xmq80