Title-1: Deep Learning Based Traffic Speed Prediction Using Long Short Term Memory for Improving Accuracy Compared with K-Nearest Neighbors Algorithm

Code:

import pandas as pd

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

from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import mean\_squared\_error

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

import numpy as np

import matplotlib.pyplot as plt

# Load your dataset

file\_path = 'your\_dataset.csv' # Replace with the actual path to your dataset

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

# Assuming your dataset has columns like 'Year', 'Month', 'Speed', 'CarCount', 'BikeCount', 'BusCount', 'TruckCount', 'Total'

# Adjust columns accordingly

features = df[['Year', 'Month', 'CarCount', 'BikeCount', 'BusCount', 'TruckCount', 'Total']]

target = df['Speed']

# Split the data into training and testing sets

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

# K-Nearest Neighbors

knn\_model = KNeighborsRegressor(n\_neighbors=5) # You can adjust the number of neighbors

knn\_model.fit(X\_train, y\_train)

knn\_predictions = knn\_model.predict(X\_test)

knn\_rmse = np.sqrt(mean\_squared\_error(y\_test, knn\_predictions))

knn\_accuracy = 1 - knn\_rmse / np.mean(y\_test)

print(f"KNN Accuracy: {knn\_accuracy \* 100:.2f}%")

# Long Short-Term Memory (LSTM)

# Normalize the data using MinMaxScaler

scaler = MinMaxScaler()

scaled\_features = scaler.fit\_transform(features)

# Reshape the data for LSTM (assuming 6 frames as input)

X\_lstm = np.array([scaled\_features[i-6:i] for i in range(6, len(scaled\_features))])

y\_lstm = np.array(target[6:])

# Split the LSTM data into training and testing sets

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

# Build the LSTM model

lstm\_model = Sequential()

lstm\_model.add(LSTM(50, input\_shape=(X\_lstm\_train.shape[1], X\_lstm\_train.shape[2])))

lstm\_model.add(Dense(1))

lstm\_model.compile(optimizer='adam', loss='mse')

# Train the LSTM model

lstm\_model.fit(X\_lstm\_train, y\_lstm\_train, epochs=10, batch\_size=32, verbose=1)

# Evaluate the LSTM model

lstm\_predictions = lstm\_model.predict(X\_lstm\_test)

lstm\_rmse = np.sqrt(mean\_squared\_error(y\_lstm\_test, lstm\_predictions))

lstm\_accuracy = 1 - lstm\_rmse / np.mean(y\_lstm\_test)

print(f"LSTM Accuracy: {lstm\_accuracy \* 100:.2f}%")

# Bar graph comparing accuracies

models = ['KNN', 'LSTM']

accuracies = [knn\_accuracy, lstm\_accuracy]

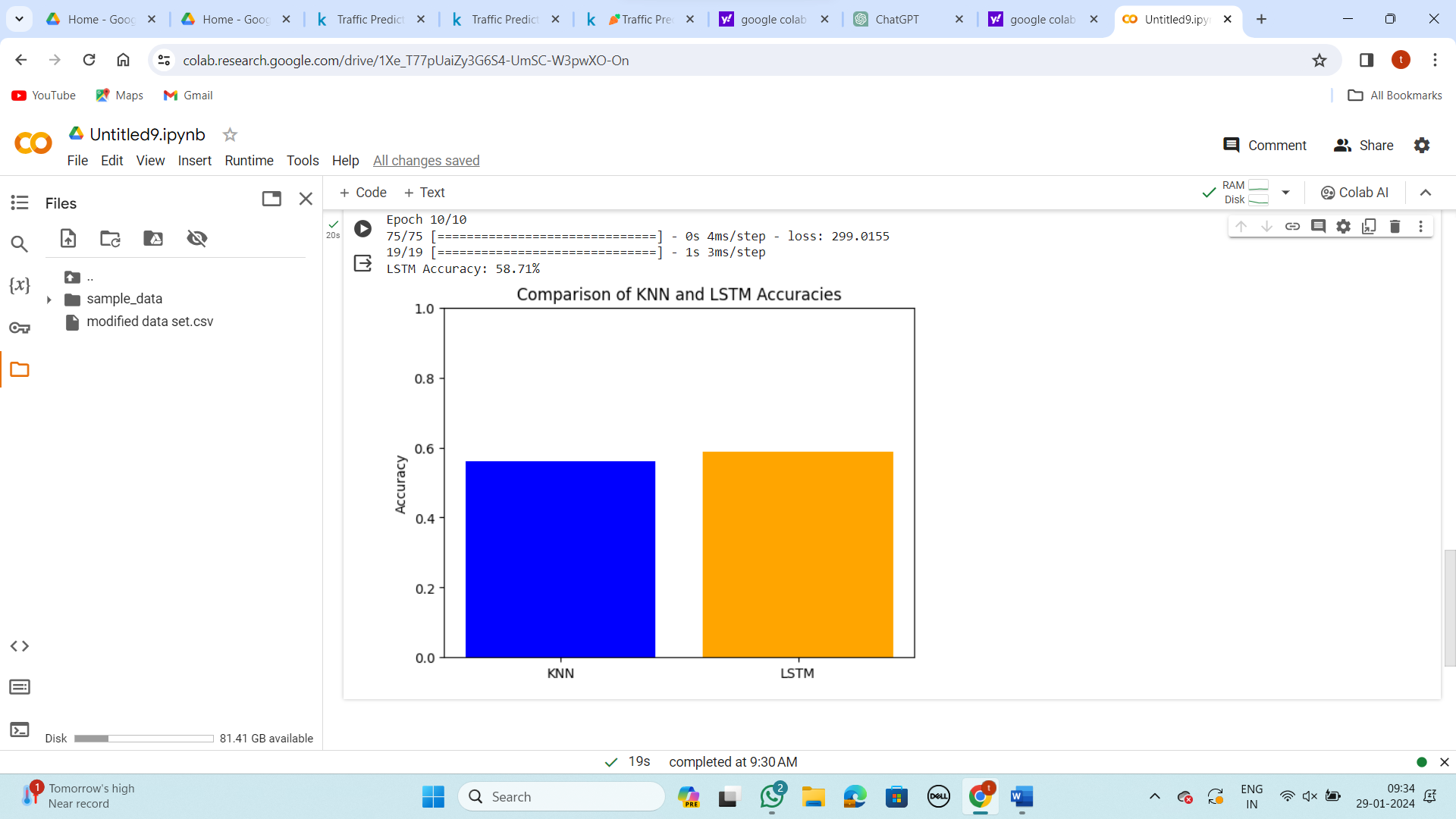
plt.bar(models, accuracies, color=['blue', 'orange'])

plt.ylabel('Accuracy')

plt.title('Comparison of KNN and LSTM Accuracies')

plt.ylim(0, 1) # Set the y-axis limit to 0-1 for percentage

plt.show()



Title-2: Deep Learning Based Traffic Speed Prediction Using Long Short Term Memory for Improving Accuracy Compared with Support Vector Regression Algorithm

Code:

import pandas as pd

import numpy as np

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

from sklearn.svm import SVR

from sklearn.metrics import mean\_squared\_error

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

import matplotlib.pyplot as plt

# Load your dataset

file\_path = 'your\_dataset.csv' # Replace with the actual path to your dataset

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

# Select relevant features

features = df[['Year', 'Month', 'CarCount', 'BikeCount', 'BusCount', 'TruckCount', 'Total']]

target = df['Speed']

# Normalize the data

scaler = MinMaxScaler()

scaled\_features = scaler.fit\_transform(features)

# Reshape the data for LSTM (assuming 6 frames as input)

X\_lstm = np.array([scaled\_features[i-6:i] for i in range(6, len(scaled\_features))])

y\_lstm = np.array(target[6:])

# Split the data into training and testing sets

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

# Support Vector Regressor (SVR)

svr\_model = SVR()

svr\_model.fit(X\_train.reshape(X\_train.shape[0], -1), y\_train)

svr\_predictions = svr\_model.predict(X\_test.reshape(X\_test.shape[0], -1))

svr\_rmse = np.sqrt(mean\_squared\_error(y\_test, svr\_predictions))

svr\_accuracy = 1 - svr\_rmse / np.mean(y\_test)

# LSTM

lstm\_model = Sequential()

lstm\_model.add(LSTM(50, input\_shape=(X\_train.shape[1], X\_train.shape[2])))

lstm\_model.add(Dense(1))

lstm\_model.compile(optimizer='adam', loss='mse')

lstm\_model.fit(X\_train, y\_train, epochs=10, batch\_size=32, verbose=1)

lstm\_predictions = lstm\_model.predict(X\_test)

lstm\_rmse = np.sqrt(mean\_squared\_error(y\_test, lstm\_predictions))

lstm\_accuracy = 1 - lstm\_rmse / np.mean(y\_test)

# Comparison and Visualization

models = ['SVR', 'LSTM']

accuracies = [svr\_accuracy, lstm\_accuracy]

plt.bar(models, accuracies, color=['blue', 'green'])

plt.ylabel('Accuracy')

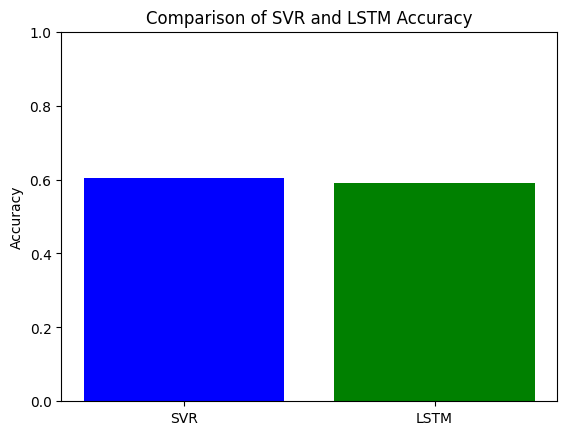
plt.title('Comparison of SVR and LSTM Accuracy')

plt.ylim(0, 1) # Assuming accuracy is between 0 and 1

plt.show()

print(f"SVR Accuracy: {svr\_accuracy \* 100:.2f}%")

print(f"LSTM Accuracy: {lstm\_accuracy \* 100:.2f}%")



Title-3: Deep Learning Based Traffic Speed Prediction Using Long Short Term Memory for Improving Accuracy Compared with Decision Tree Algorithm

Code:

import pandas as pd

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

from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import mean\_squared\_error

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

import numpy as np

import matplotlib.pyplot as plt

# Load your dataset

file\_path = 'your\_dataset.csv' # Replace with the actual path to your dataset

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

# Assuming your dataset has columns like 'Year', 'Month', 'Speed', 'CarCount', 'BikeCount', 'BusCount', 'TruckCount', 'Total'

# Adjust columns accordingly

features = df[['Year', 'Month', 'CarCount', 'BikeCount', 'BusCount', 'TruckCount', 'Total']]

target = df['Speed']

# Split the data into training and testing sets

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

# Decision Tree

decision\_tree\_model = DecisionTreeRegressor(random\_state=42)

decision\_tree\_model.fit(X\_train, y\_train)

decision\_tree\_predictions = decision\_tree\_model.predict(X\_test)

decision\_tree\_rmse = np.sqrt(mean\_squared\_error(y\_test, decision\_tree\_predictions))

decision\_tree\_accuracy = 1 - decision\_tree\_rmse / np.mean(y\_test)

# LSTM

# Normalize the data using MinMaxScaler

scaler = MinMaxScaler()

scaled\_features = scaler.fit\_transform(features)

# Reshape the data for LSTM (assuming 6 frames as input)

X\_lstm = np.array([scaled\_features[i-6:i] for i in range(6, len(scaled\_features))])

y\_lstm = np.array(target[6:])

# Split the LSTM data into training and testing sets

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

# Build the LSTM model

lstm\_model = Sequential()

lstm\_model.add(LSTM(50, input\_shape=(X\_lstm\_train.shape[1], X\_lstm\_train.shape[2])))

lstm\_model.add(Dense(1))

lstm\_model.compile(optimizer='adam', loss='mse')

# Train the LSTM model

lstm\_model.fit(X\_lstm\_train, y\_lstm\_train, epochs=10, batch\_size=32, verbose=1)

# Evaluate the LSTM model

lstm\_predictions = lstm\_model.predict(X\_lstm\_test)

lstm\_rmse = np.sqrt(mean\_squared\_error(y\_lstm\_test, lstm\_predictions))

lstm\_accuracy = 1 - lstm\_rmse / np.mean(y\_lstm\_test)

# Display accuracies

print(f"Decision Tree Accuracy: {decision\_tree\_accuracy}")

print(f"LSTM Accuracy: {lstm\_accuracy}")

# Plotting the results

labels = ['Decision Tree', 'LSTM']

accuracies = [decision\_tree\_accuracy \* 100, lstm\_accuracy \* 100]

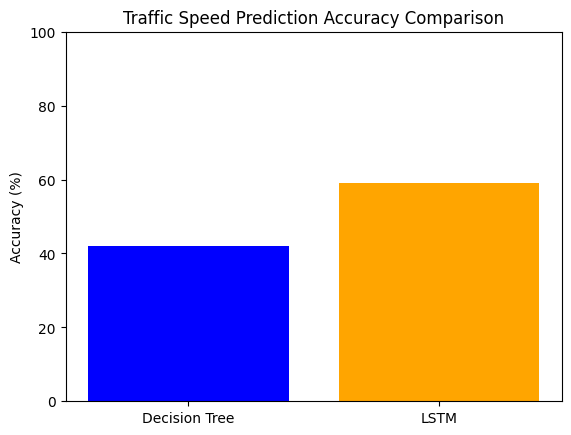
plt.bar(labels, accuracies, color=['blue', 'orange'])

plt.ylabel('Accuracy (%)')

plt.title('Traffic Speed Prediction Accuracy Comparison')

plt.ylim(0, 100)

plt.show()



Title-4: Deep Learning Based Traffic Speed Prediction Using Long Short Term Memory for Improving Accuracy Compared with Linear Regression Algorithm

code:

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

import numpy as np

import matplotlib.pyplot as plt

# Load your dataset

file\_path = 'your\_dataset.csv' # Replace with the actual path to your dataset

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

# Extract relevant features

features = df[['Year', 'Month', 'CarCount', 'BikeCount', 'BusCount', 'TruckCount', 'Total']]

target = df['Speed']

# Split the data into training and testing sets

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

# Linear Regression

lr\_model = LinearRegression()

lr\_model.fit(X\_train, y\_train)

lr\_predictions = lr\_model.predict(X\_test)

lr\_rmse = np.sqrt(mean\_squared\_error(y\_test, lr\_predictions))

lr\_accuracy = 1 - lr\_rmse / np.mean(y\_test)

print(f"Linear Regression Accuracy: {lr\_accuracy}")

# Long Short-Term Memory (LSTM)

# Normalize the data using MinMaxScaler

scaler = MinMaxScaler()

scaled\_features = scaler.fit\_transform(features)

# Reshape the data for LSTM (assuming 6 frames as input)

X\_lstm = np.array([scaled\_features[i-6:i] for i in range(6, len(scaled\_features))])

y\_lstm = np.array(target[6:])

# Split the LSTM data into training and testing sets

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

# Build the LSTM model

lstm\_model = Sequential()

lstm\_model.add(LSTM(50, input\_shape=(X\_lstm\_train.shape[1], X\_lstm\_train.shape[2])))

lstm\_model.add(Dense(1))

lstm\_model.compile(optimizer='adam', loss='mse')

# Train the LSTM model

lstm\_model.fit(X\_lstm\_train, y\_lstm\_train, epochs=10, batch\_size=32, verbose=1)

# Evaluate the LSTM model

lstm\_predictions = lstm\_model.predict(X\_lstm\_test)

lstm\_rmse = np.sqrt(mean\_squared\_error(y\_lstm\_test, lstm\_predictions))

lstm\_accuracy = 1 - lstm\_rmse / np.mean(y\_lstm\_test)

print(f"LSTM Accuracy: {lstm\_accuracy}")

# Compare accuracies using a bar graph

models = ['Linear Regression', 'LSTM']

accuracies = [lr\_accuracy, lstm\_accuracy]

plt.bar(models, accuracies, color=['blue', 'orange'])

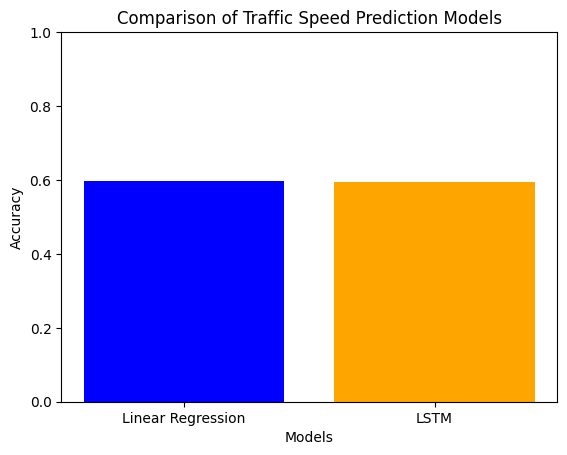
plt.xlabel('Models')

plt.ylabel('Accuracy')

plt.title('Comparison of Traffic Speed Prediction Models')

plt.ylim(0, 1) # Assuming accuracy is in the range [0, 1]

plt.show()



Main title: Deep Learning Based Traffic Speed Prediction Using Long Short Term Memory for Improving Accuracy Compared with LSTM Algorithm

CODE:

import pandas as pd

import numpy as np

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

import matplotlib.pyplot as plt

# Load your dataset

file\_path = 'your\_dataset.csv' # Replace with the actual path to your dataset

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

# Extract relevant features

features = df[['Year', 'Month', 'CarCount', 'BikeCount', 'BusCount', 'TruckCount', 'Total']]

target = df['Speed']

# Split the data into training and testing sets

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

# Linear Regression Model

linear\_model = LinearRegression()

linear\_model.fit(X\_train, y\_train)

linear\_predictions = linear\_model.predict(X\_test)

linear\_rmse = np.sqrt(mean\_squared\_error(y\_test, linear\_predictions))

linear\_accuracy = 1 - linear\_rmse / np.mean(y\_test)

print(f"Linear Regression Accuracy: {linear\_accuracy \* 100:.2f}%")

# LSTM Model

scaler = MinMaxScaler()

scaled\_features = scaler.fit\_transform(features)

X\_lstm = np.array([scaled\_features[i-6:i] for i in range(6, len(scaled\_features))])

y\_lstm = np.array(target[6:])

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

lstm\_model = Sequential()

lstm\_model.add(LSTM(50, input\_shape=(X\_lstm\_train.shape[1], X\_lstm\_train.shape[2])))

lstm\_model.add(Dense(1))

lstm\_model.compile(optimizer='adam', loss='mse')

lstm\_model.fit(X\_lstm\_train, y\_lstm\_train, epochs=10, batch\_size=32, verbose=1)

lstm\_predictions = lstm\_model.predict(X\_lstm\_test)

lstm\_rmse = np.sqrt(mean\_squared\_error(y\_lstm\_test, lstm\_predictions))

lstm\_accuracy = 1 - lstm\_rmse / np.mean(y\_lstm\_test)

print(f"LSTM Accuracy: {lstm\_accuracy \* 100:.2f}%")

# Plotting the comparison

labels = ['Linear Regression', 'LSTM']

accuracies = [linear\_accuracy \* 100, lstm\_accuracy \* 100]

plt.bar(labels, accuracies, color=['blue', 'orange'])

plt.ylabel('Accuracy (%)')

plt.title('Traffic Speed Prediction Accuracy Comparison')

plt.show()

