MODEL USED : FACEBOOK PROPHET

I used Facebook Prophet for time series forecasting.

WHY PROPHET?

"For forecasting daily passenger journeys, I chose Facebook Prophet because it is designed for time series data with multiple seasonality patterns, such as weekly and daily cycles. Prophet handles missing data and outliers well, which is common in transport data. It requires minimal manual tuning, making it efficient for rapid prototyping. Compared to classical models like ARIMA, which needs stationary data and complex parameter tuning, Prophet is more flexible. Deep learning models like LSTM require more data and computational resources, which aren't ideal here. Thus, Prophet offers the best balance of accuracy, interpretability, and ease of use for this business problem."

```
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

# Load dataset
df = pd.read_csv('Daily_Public_Transport.csv')


# Convert 'Date' to datetime
df['Date'] = pd.to_datetime(df['Date'], dayfirst=True)

# Remove commas from number columns and convert to float
services = ['Local Route', 'Light Rail', 'Peak Service', 'Rapid Route', 'School']

for col in services:
    df[col] = df[col].astype(str).str.replace(',', '').astype(float)

# Check data types and nulls
print(df[services].info())
print(df.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1918 entries, 0 to 1917
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Local Route   1918 non-null   float64
 1   Light Rail    1918 non-null   float64
 2   Peak Service  1918 non-null   float64
 3   Rapid Route   1918 non-null   float64
 4   School        1918 non-null   float64
dtypes: float64(5)
memory usage: 75.1 KB
None
Date             0
Local Route      0
Light Rail       0
Peak Service     0
Rapid Route      0
School           0
Other           20
dtype: int64
```

```
forecasts = {}

for service in services:
    # Prepare dataframe for Prophet
    prophet_df = df[['Date', service]].rename(columns={'Date': 'ds', service: 'y'})

    # Initialize and fit Prophet model with daily + weekly seasonality
    model = Prophet(daily_seasonality=True, weekly_seasonality=True)
    model.fit(prophet_df)

    # Make future dataframe for next 7 days
    future = model.make_future_dataframe(periods=7)
```

```
# Predict
forecast = model.predict(future)

# Keep only future 7 days forecasts and clip negative values to zero
future_forecast = forecast[forecast['ds'] > prophet_df['ds'].max()]
future_forecast['yhat'] = future_forecast['yhat'].clip(lower=0)
future_forecast['yhat_lower'] = future_forecast['yhat_lower'].clip(lower=0)
future_forecast['yhat_upper'] = future_forecast['yhat_upper'].clip(lower=0)

# Store forecast
forecasts[service] = future_forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']]
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
  future_forecast['yhat_lower'] = future_forecast['yhat_lower'].clip(lower=0)
<ipython-input-8-47b78460e6c4>:21: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
  future_forecast['yhat_upper'] = future_forecast['yhat_upper'].clip(lower=0)
DEBUG:cmdstanpy:input tempfile: /tmp/tmpqoeeveoa/sblgz4ad.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpqoeeveoa/fun3lo4u.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.11/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=63197'
06:07:42 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
06:07:42 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<ipython-input-8-47b78460e6c4>:19: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
  future_forecast['yhat'] = future_forecast['yhat'].clip(lower=0)
<ipython-input-8-47b78460e6c4>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
  future_forecast['yhat_lower'] = future_forecast['yhat_lower'].clip(lower=0)
<ipython-input-8-47b78460e6c4>:21: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
  future_forecast['yhat_upper'] = future_forecast['yhat_upper'].clip(lower=0)
DEBUG:cmdstanpy:input tempfile: /tmp/tmpqoeeveoa/9me1x0ga.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmpqoeeveoa/0_kos6at.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.11/dist-packages/prophet/stan_model/prophet_model.bin', 'random', 'seed=58485'
06:07:43 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
06:07:43 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
<ipython-input-8-47b78460e6c4>:19: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
  future_forecast['yhat'] = future_forecast['yhat'].clip(lower=0)
<ipython-input-8-47b78460e6c4>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a
  future_forecast['yhat_lower'] = future_forecast['yhat_lower'].clip(lower=0)
<ipython-input-8-47b78460e6c4>:21: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

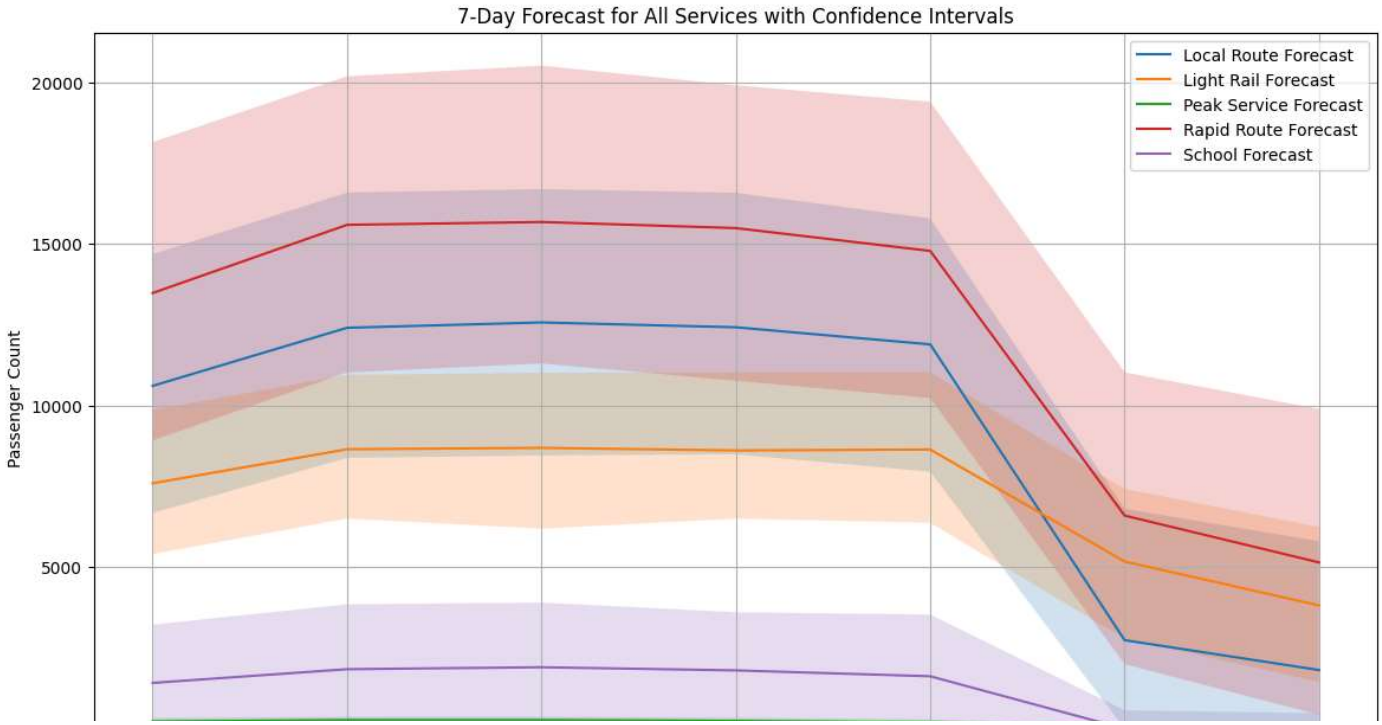
```
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

# Load dataset
df = pd.read_csv('Daily_Public_Transport.csv')

# Convert 'Date' to datetime with dayfirst=True to fix date parsing warning
```

```
df['Date'] = pd.to_datetime(df['Date'], dayfirst=True)

# Remove commas and convert all service columns to numeric
service_types = ['Local Route', 'Light Rail', 'Peak Service', 'Rapid Route', 'School']
for service in service_types:
    df[service] = df[service].astype(str).str.replace(',', '').astype(float)

# Dictionary to store forecasts
forecasts = {}

for service in service_types:
    # Prepare data for Prophet
    prophet_df = df[['Date', service]].rename(columns={'Date': 'ds', service: 'y'})

    # Fit Prophet model
    model = Prophet(daily_seasonality=True, weekly_seasonality=True)
    model.fit(prophet_df)

    # Forecast next 7 days
    future = model.make_future_dataframe(periods=7)
    forecast = model.predict(future)

    # Clip predictions to no negatives
    forecast['yhat'] = forecast['yhat'].clip(lower=0)
    forecast['yhat_lower'] = forecast['yhat_lower'].clip(lower=0)
    forecast['yhat_upper'] = forecast['yhat_upper'].clip(lower=0)

    # Store forecast
    forecasts[service] = forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']]

    # Plot forecast
    plt.figure(figsize=(10, 5))
    model.plot(forecast)
    plt.title(f'7-Day Forecast for {service}')
    plt.xlabel('Date')
    plt.ylabel('Passenger Count')
    plt.show()

# Combine next 7 days forecasts into one table with original service names
next_7_days = forecasts[service_types[0]][['ds']].copy()
for service in service_types:
    next_7_days[service] = forecasts[service]['yhat'].values[-7:]

# Rename columns to match original dataset format
next_7_days.rename(columns={'ds': 'Date'}, inplace=True)

print("Next 7 Days Forecasted Passenger Counts:")
print(next_7_days)

# --- Model Evaluation on last 7 days ---

eval_results = {}

for service in service_types:
    # Prepare data
    prophet_df = df[['Date', service]].rename(columns={'Date': 'ds', service: 'y'})

    # Train-test split: last 7 days as test
    train = prophet_df[:-7]
    test = prophet_df[-7:]

    # Fit model on train data
    model = Prophet(daily_seasonality=True, weekly_seasonality=True)
    model.fit(train)

    # Predict on test period
    future = model.make_future_dataframe(periods=7)
    forecast = model.predict(future)

    # Extract forecast for test dates
    forecast_test = forecast.set_index('ds').loc[test['ds']]

    # Clip predictions to no negatives
    y_pred = np.clip(forecast_test['yhat'].values, a_min=0, a_max=None)
    y_true = test['y'].values

    # Calculate metrics
```

```
    mae = mean_absolute_error(y_true, y_pred)
    rmse = mean_squared_error(y_true, y_pred, squared=False)
    mape = np.mean(np.abs((y_true - y_pred) / y_true)) * 100

    eval_results[service] = {'MAE': mae, 'RMSE': rmse, 'MAPE (%)': mape}

# Display evaluation metrics
eval_df = pd.DataFrame(eval_results).T
print("\nModel Evaluation Metrics (Last 7 Days):")
print(eval_df)



# Create a combined dataframe with dates and all services
forecast_days = forecasts[services[0]]['ds'].reset_index(drop=True)

forecast_table = pd.DataFrame({'Date': forecast_days})

for service in services:
    forecast_table[service] = forecasts[service]['yhat'].values.round().astype(int)

print(forecast_table)
```

```
         Date  Local Route  Light Rail  Peak Service  Rapid Route  School
0 2024-09-30        10605        7591           224        13476    1418
1 2024-10-01        12401        8645           270        15584    1842
2 2024-10-02        12566        8687           270        15672    1903
3 2024-10-03        12416        8606           243        15484    1805
4 2024-10-04        11888        8636           191        14778    1623
5 2024-10-05         2740        5172             0         6593       0
6 2024-10-06         1816        3815             0         5145       0
```

```
plt.figure(figsize=(14,8))

for service in services:
    forecast_df = forecasts[service]
    plt.plot(forecast_df['ds'], forecast_df['yhat'], label=f'{service} Forecast')
    plt.fill_between(forecast_df['ds'], forecast_df['yhat_lower'], forecast_df['yhat_upper'], alpha=0.2)

plt.title('7-Day Forecast for All Services with Confidence Intervals')
plt.xlabel('Date')
plt.ylabel('Passenger Count')
plt.legend()
plt.grid(True)
plt.show()
```


7-Day Forecast for All Services with Confidence Intervals
Passenger Count
20000
15000
10000
5000
Local Route Forecast
Light Rail Forecast
Peak Service Forecast
Rapid Route Forecast
School Forecast