

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error, r2_score
from sklearn.ensemble import RandomForestRegressor

import ipywidgets as widgets
from IPython.display import display
```

```
import os
os.listdir("/content")
```

```
['.config',
 '.ipynb_checkpoints',
 'traffic_volume_interstate.xlsx',
 'sample_data']
```

```
import pandas as pd

df = pd.read_excel("/content/traffic_volume_interstate.xlsx")

df.head()
```

	traffic_volume	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	date_time
0	5545	NaN	288.28	0.0	0.0	40	Clouds	scattered clouds	2012-02-10 09:00:00
1	4516	NaN	289.36	0.0	0.0	75	Clouds	broken clouds	2012-02-10 10:00:00
2	4767	NaN	289.58	0.0	0.0	90	Clouds	overcast clouds	2012-02-10 11:00:00
3	5026	NaN	290.13	0.0	0.0	90	Clouds	overcast clouds	2012-02-10 12:00:00
4	4918	NaN	291.14	0.0	0.0	75	Clouds	broken clouds	2012-02-10 13:00:00

Next steps: [Generate code with df](#) [New interactive sheet](#)

```
df['date_time'] = pd.to_datetime(df['date_time'])
```

```
df['hour'] = df['date_time'].dt.hour      # 0-23
df['day'] = df['date_time'].dt.day        # 1-31
df['month'] = df['date_time'].dt.month    # 1-12
df['weekday'] = df['date_time'].dt.weekday # 0 = Monday, 6 = Sunday
```

```
df['is_weekend'] = df['weekday'].apply(lambda x: 1 if x >= 5 else 0)
```

```
df = df.drop(columns=['date_time', 'weather_description'])
```

```
df.head()
```

	traffic_volume	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	hour	day	month	weekday	is_weekend
0	5545	NaN	288.28	0.0	0.0	40	Clouds	9	10	2	4	0
1	4516	NaN	289.36	0.0	0.0	75	Clouds	10	10	2	4	0
2	4767	NaN	289.58	0.0	0.0	90	Clouds	11	10	2	4	0
3	5026	NaN	290.13	0.0	0.0	90	Clouds	12	10	2	4	0
4	4918	NaN	291.14	0.0	0.0	75	Clouds	13	10	2	4	0

Next steps: [Generate code with df](#) [New interactive sheet](#)

Start coding or [generate](#) with AI.

```
X = df.drop('traffic_volume', axis=1)
y = df['traffic_volume']
```

```
categorical_features = ['holiday', 'weather_main']
numerical_features = [
    'temp', 'rain_1h', 'snow_1h', 'clouds_all',
    'hour', 'day', 'month', 'weekday', 'is_weekend'
]
```

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features),
        ('num', 'passthrough', numerical_features)
    ]
)
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.pipeline import Pipeline

model = RandomForestRegressor(
    n_estimators=150,
    random_state=42,
    n_jobs=-1
)

pipeline = Pipeline(steps=[
    ('preprocess', preprocessor),
    ('model', model)
])
```

```
print(pipeline)
```

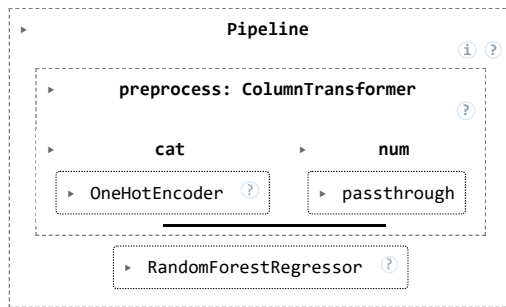
```
Pipeline(steps=[('preprocess',
  ColumnTransformer(transformers=[('cat',
    OneHotEncoder(handle_unknown='ignore'),
    ['holiday', 'weather_main']),
    ('num', 'passthrough',
    ['temp', 'rain_1h', 'snow_1h',
    'clouds_all', 'hour', 'day',
    'month', 'weekday',
    'is_weekend'])])),
  ('model',
    RandomForestRegressor(n_estimators=150, n_jobs=-1,
    random_state=42))])
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
```

```
)
```

```
pipeline.fit(X_train, y_train)
```



```
from sklearn.metrics import mean_absolute_error, r2_score
```

```
y_pred = pipeline.predict(X_test)
```

```
print("MAE:", mean_absolute_error(y_test, y_pred))
```

```
print("R2 :", r2_score(y_test, y_pred))
```

```
MAE: 324.3144452511842
```

```
R2 : 0.9184944270921682
```

```
import ipywidgets as widgets
from IPython.display import display
```

```

date_input = widgets.Text(
    description='DateTime:',
    placeholder='YYYY-MM-DD HH:MM'
)

temp_input = widgets.FloatText(description='Temp:')
rain_input = widgets.FloatText(description='Rain:')
snow_input = widgets.FloatText(description='Snow:')
clouds_input = widgets.IntText(description='Clouds %')

holiday_input = widgets.Dropdown(
    options=['None', 'Christmas Day', 'New Years Day'],
    description='Holiday:'
)

weather_input = widgets.Dropdown(
    options=['Clear', 'Clouds', 'Rain', 'Snow', 'Mist'],
    description='Weather:'
)

button = widgets.Button(description="Predict Traffic")
output = widgets.Output()

```

```

def predict_traffic(b):
    with output:
        output.clear_output()

        dt = pd.to_datetime(date_input.value)




        input_df = pd.DataFrame([
            'holiday': holiday_input.value,
            'weather_main': weather_input.value,
            'temp': temp_input.value,
            'rain_1h': rain_input.value,
            'snow_1h': snow_input.value,
            'clouds_all': clouds_input.value,
            'hour': dt.hour,
            'day': dt.day,

```

```
        'month': dt.month,  
        'weekday': dt.weekday(),  
        'is_weekend': 1 if dt.weekday() >= 5 else 0  
    })  
  
    prediction = pipeline.predict(input_df)[0]  
  
    print(f"🚗 Estimated Traffic Volume: {int(prediction)}")
```

```
button.on_click(predict_traffic)
```

```
rf_model = pipeline.named_steps['model']  
  
feature_names_cat = pipeline.named_steps['preprocess'].transformers_[0][1].get_feature_names_out(categorical_features)  
feature_names_num = numerical_features  
feature_names = list(feature_names_cat) + feature_names_num  
  
importances = rf_model.feature_importances_  
  
fi_df = pd.DataFrame({  
    'feature': feature_names,  
    'importance': importances  
}).sort_values(by='importance', ascending=False)  
  
fi_df
```

	feature	importance	
27	hour	8.013134e-01	
28	day	4.927362e-02	
23	temp	4.536245e-02	
30	weekday	3.673345e-02	
29	month	2.337099e-02	
31	is_weekend	1.787667e-02	
26	clouds_all	1.044792e-02	
24	rain_1h	4.205795e-03	
13	weather_main_Clouds	3.033156e-03	
12	weather_main_Clear	1.831370e-03	
17	weather_main_Mist	1.442944e-03	
15	weather_main_Fog	1.049448e-03	
20	weather_main_Snow	1.006200e-03	
16	weather_main_Haze	9.942554e-04	
18	weather_main_Rain	9.838999e-04	
22	weather_main_Thunderstorm	4.647165e-04	
14	weather_main_Drizzle	4.329293e-04	
19	weather_main_Smoke	9.385329e-05	
25	snow_1h	5.297104e-05	
6	holiday_New Years Day	1.121584e-05	
5	holiday_Memorial Day	7.140101e-06	
11	holiday_nan	5.164131e-06	
21	weather_main_Squall	2.754218e-06	
0	holiday_Christmas Day	1.387744e-06	
2	holiday_Independence Day	5.845815e-07	
3	holiday_Labor Day	5.102860e-07	
9	holiday_Veterans Day	4.917608e-07	
8	holiday_Thanksgiving Day	3.955932e-07	
1	holiday_Columbus Day	1.700547e-07	
10	holiday_Washingtons Birthday	8.440443e-08	
4	holiday_Martin Luther King Jr Day	5.334930e-08	
7	holiday_State Fair	1.443799e-08	

Next steps:

[Generate code with fi_df](#)[New interactive sheet](#)

```

import matplotlib.pyplot as plt
import seaborn as sns

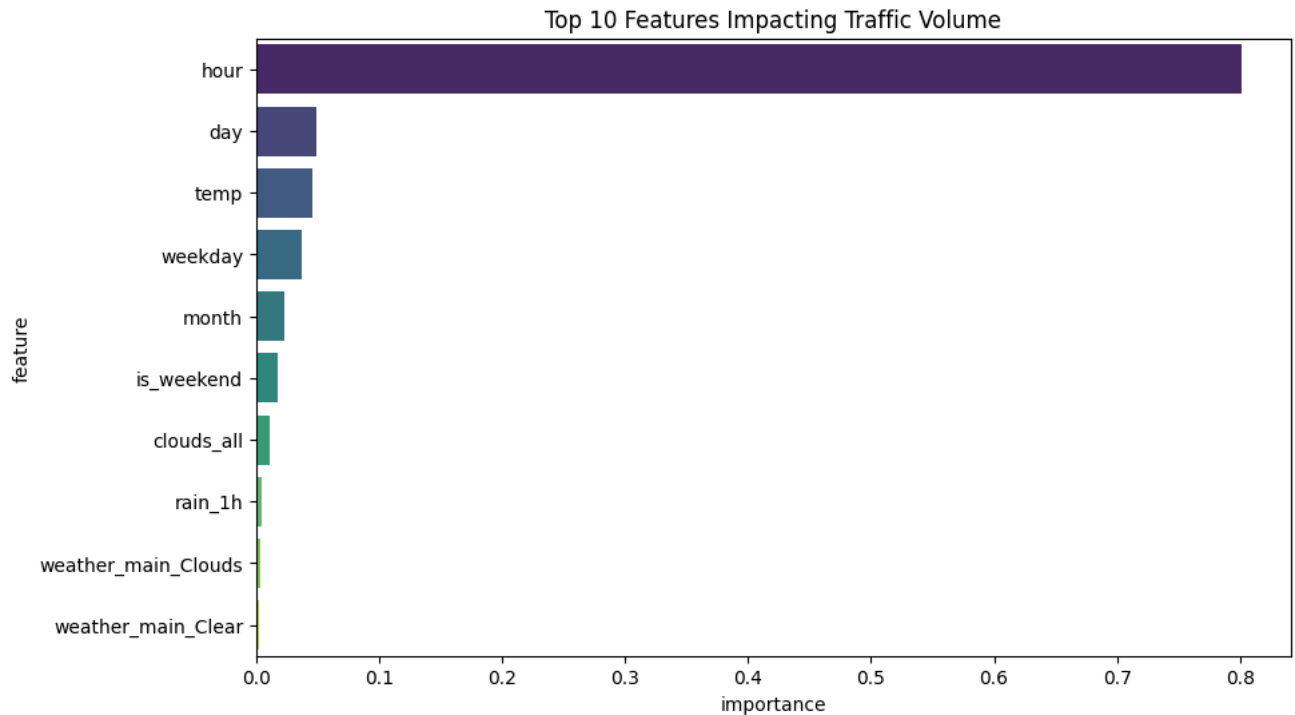
plt.figure(figsize=(10,6))
sns.barplot(data=fi_df.head(10), x='importance', y='feature', palette='viridis')
plt.title("Top 10 Features Impacting Traffic Volume")
plt.show()

```

```
/tmp/ipython-input-3180554061.py:5: FutureWarning:
```

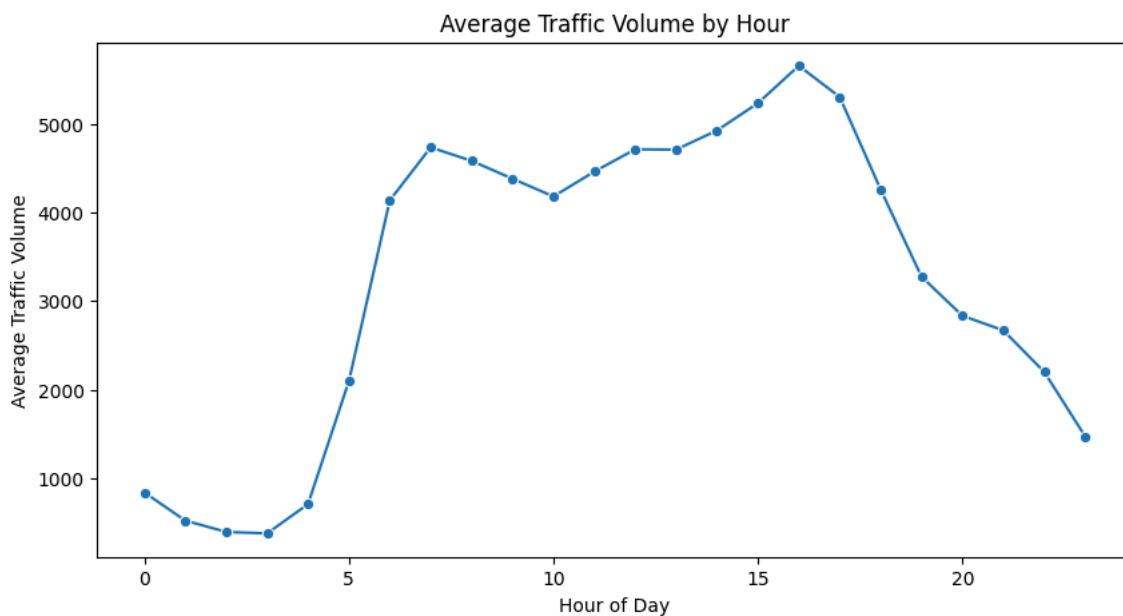
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set

```
sns.barplot(data=fi_df.head(10), x='importance', y='feature', palette='viridis')
```



```
hourly = df.groupby('hour')['traffic_volume'].mean().reset_index()
```

```
plt.figure(figsize=(10,5))
sns.lineplot(data=hourly, x='hour', y='traffic_volume', marker='o')
plt.title("Average Traffic Volume by Hour")
plt.xlabel("Hour of Day")
plt.ylabel("Average Traffic Volume")
plt.show()
```



```
holiday_avg = df.groupby('holiday')['traffic_volume'].mean().reset_index()
```

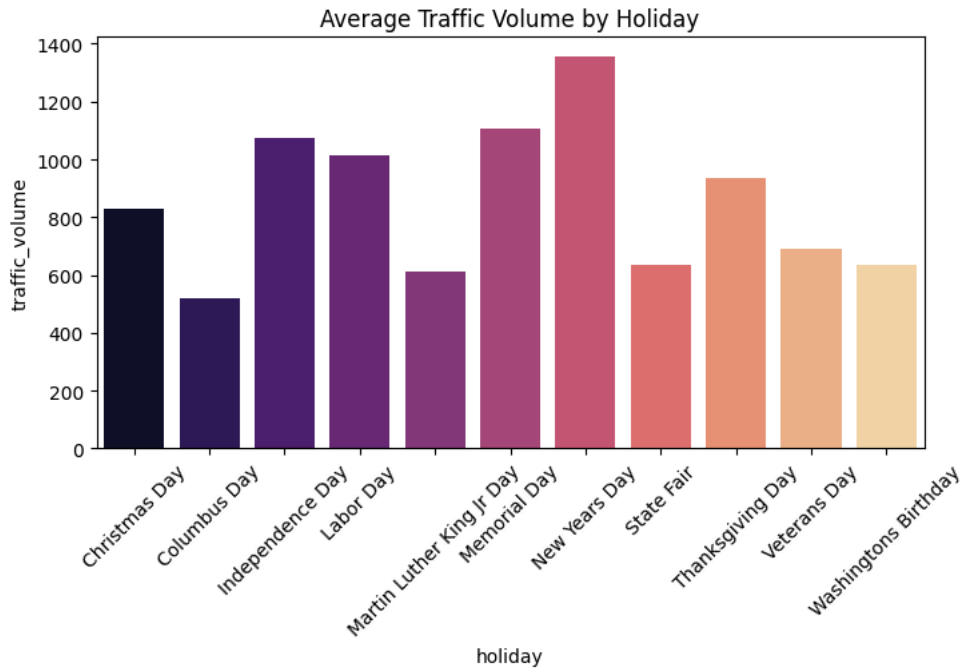
```
plt.figure(figsize=(8,4))
sns.barplot(data=holiday_avg, x='holiday', y='traffic_volume', palette='magma')
plt.title("Average Traffic Volume by Holiday")
```

```
plt.xticks(rotation=45)
plt.show()
```

/tmp/ipython-input-2903217567.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

```
sns.barplot(data=holiday_avg, x='holiday', y='traffic_volume', palette='magma')
```



```
weekend_avg = df.groupby('is_weekend')['traffic_volume'].mean().reset_index()
```

```
plt.figure(figsize=(5,4))
sns.barplot(data=weekend_avg, x='is_weekend', y='traffic_volume', palette='coolwarm')
plt.title("Average Traffic Volume: Weekday vs Weekend")
plt.xticks([0,1], ['Weekday', 'Weekend'])
plt.show()
```

/tmp/ipython-input-1291450864.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

```
sns.barplot(data=weekend_avg, x='is_weekend', y='traffic_volume', palette='coolwarm')
```

