In [2]:
```
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
import numpy as np
from statsmodels.tsa.seasonal import seasonal_decompose
from sklearn.preprocessing import StandardScaler
```

In [21]:
```
# df = pd.read_csv('train_clean_v2.csv')
# df.drop([df.columns[0]],inplace=True,axis=1)
# df.head()
```

Out[21]:

	Timestamp	% Baseline	maxtempC	mintempC	sunHour	uvIndex	DewPointC	FeelsLikeC
0	2014-01-01 07:00:00	0.0079	-3	-6	8.7	2	-14	-13
1	2014-01-01 08:00:00	0.1019	-3	-6	8.7	2	-14	-12
2	2014-01-01 09:00:00	0.3932	-3	-6	8.7	2	-14	-11
3	2014-01-01 10:00:00	0.5447	-3	-6	8.7	2	-14	-10
4	2014-01-01 11:00:00	0.5485	-3	-6	8.7	2	-14	-10

5 rows × 25 columns

In [22]:
```
# # Load the trained .h5 model
# from tensorflow.keras.models import load_model
# model = load_model('ann_modelv2.h5')

# # Identify rows with NaN in % Baseline
# nan_rows = df[df['% Baseline'].isna()]

# # Prepare the features (excluding Timestamp and % Baseline)
# features = nan_rows.drop(columns=['Timestamp', '% Baseline'])

# # Scale the features (ensure the same scaler used during training is applied here
# from sklearn.preprocessing import StandardScaler
# scaler = StandardScaler()
# scaled_features = scaler.fit_transform(features)

# # Make predictions using the model
# predictions = model.predict(scaled_features)

# # Fill the NaN values in the original dataframe
# df.loc[df['% Baseline'].isna(), '% Baseline'] = predictions
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be buil
t. `model.compile_metrics` will be empty until you train or evaluate the model.
570/570 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
```

In [25]:
```
# # Display the updated dataframe
# df.head()

# # Save the updated dataframe to a new CSV file
# df.to_csv('analysisv2.csv', index=False)
```

In [60]:
```
# Load the analysis.csv data
analysis = pd.read_csv('analysisv2.csv')
# Ensure the 'Timestamp' column is in datetime format
analysis['Timestamp'] = pd.to_datetime(analysis['Timestamp'])
# Extract hour and month from the 'ds' column
analysis['hour'] = analysis['Timestamp'].dt.hour
analysis['month'] = analysis['Timestamp'].dt.month

# Create the interaction term 'hour_month'
analysis['hour_month'] = analysis['hour'] * analysis['month']
# Define the cutoff date
cutoff_date = pd.Timestamp('2017-10-01 00:00:00')
analysis_df = analysis[analysis['Timestamp'] < cutoff_date]
ttest = analysis[analysis['Timestamp'] >= cutoff_date]
# Prepare the data for Prophet
# Rename the columns to 'ds' for the date/time and 'y' for the target value
analysis_df['ds'] = pd.to_datetime(analysis_df['Timestamp'])
analysis_df['y'] = analysis_df['% Baseline']
# Optional: Add cap and floor if using logistic growth
analysis_df['cap'] = 1.0  # Example cap
analysis_df['floor'] = 0.0  # Example floor
# Fit and transform the humidity and sunHour columns
scaler = StandardScaler()
analysis_df[['shumidity', 'ssunHour','scloudcover','shour_month']] = scaler.fit_tra
# Keep only the columns required by Prophet
prophet_df = analysis_df[['ds', 'y','cap','floor', 'ssunHour', 'shumidity','scloudc

split_point = int(len(analysis_df) * 0.8)
# Split the data based on the split point
train_df = prophet_df[:split_point]
test_df = prophet_df[split_point:]

# Extract the target variable '% Baseline'
#y_train = train_df['% Baseline'].values
#y_test = test_df['% Baseline'].values

# Create a DataFrame for prediction, using the 'ds' from test_df
#future_test = test_df[['ds','ssunHour','shumidity']].copy()
```

```
C:\Users\NAUFAL\AppData\Local\Temp\ipykernel_28172\3752391712.py:17: SettingWithCopy
Warning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  analysis_df['ds'] = pd.to_datetime(analysis_df['Timestamp'])
C:\Users\NAUFAL\AppData\Local\Temp\ipykernel_28172\3752391712.py:18: SettingWithCopy
Warning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  analysis_df['y'] = analysis_df['% Baseline']
C:\Users\NAUFAL\AppData\Local\Temp\ipykernel_28172\3752391712.py:20: SettingWithCopy
Warning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  analysis_df['cap'] = 1.0  # Example cap
C:\Users\NAUFAL\AppData\Local\Temp\ipykernel_28172\3752391712.py:21: SettingWithCopy
Warning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  analysis_df['floor'] = 0.0  # Example floor
C:\Users\NAUFAL\AppData\Local\Temp\ipykernel_28172\3752391712.py:24: SettingWithCopy
Warning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  analysis_df[['shumidity', 'ssunHour','scloudcover','shour_month']] = scaler.fit_tr
ansform(analysis_df[['humidity', 'sunHour','cloudcover','hour_month']])
C:\Users\NAUFAL\AppData\Local\Temp\ipykernel_28172\3752391712.py:24: SettingWithCopy
Warning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  analysis_df[['shumidity', 'ssunHour','scloudcover','shour_month']] = scaler.fit_tr
ansform(analysis_df[['humidity', 'sunHour','cloudcover','hour_month']])
C:\Users\NAUFAL\AppData\Local\Temp\ipykernel_28172\3752391712.py:24: SettingWithCopy
Warning:
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/u
```

```
ser_guide/indexing.html#returning-a-view-versus-a-copy
  analysis_df[['shumidity', 'ssunHour','scloudcover','shour_month']] = scaler.fit_tr
ansform(analysis_df[['humidity', 'sunHour','cloudcover','hour_month']])
C:\Users\NAUFAL\AppData\Local\Temp\ipykernel_28172\3752391712.py:24: SettingWithCopy
Warning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  analysis_df[['shumidity', 'ssunHour','scloudcover','shour_month']] = scaler.fit_tr
ansform(analysis_df[['humidity', 'sunHour','cloudcover','hour_month']])
```

In [61]:
```
prophet_df
```

Out[61]:

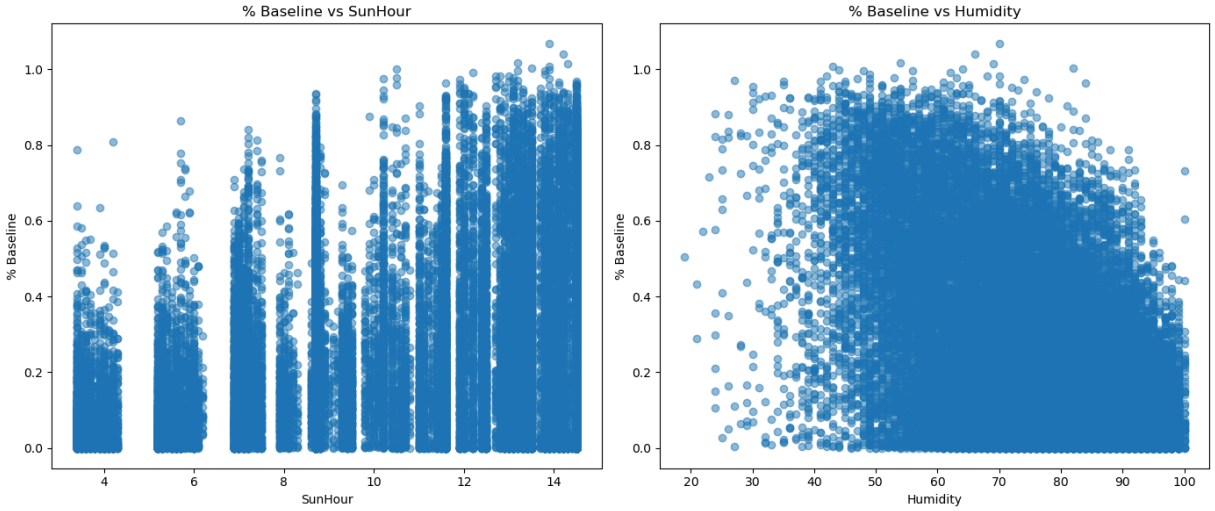
	ds	y	cap	floor	ssunHour	shumidity	scloudcover	shour_month
0	2014-01-01 07:00:00	0.007900	1.0	0.0	-0.422555	-1.681535	-0.795529	-1.037312
1	2014-01-01 08:00:00	0.101900	1.0	0.0	-0.422555	-1.888416	-0.852592	-1.021246
2	2014-01-01 09:00:00	0.393200	1.0	0.0	-0.422555	-2.095297	-0.909655	-1.005181
3	2014-01-01 10:00:00	0.544700	1.0	0.0	-0.422555	-2.095297	-0.738466	-0.989116
4	2014-01-01 11:00:00	0.548500	1.0	0.0	-0.422555	-2.164257	-0.567277	-0.973050
...	...	...	...	...	...	...	...	...
32844	2017-09-30 19:00:00	0.189902	1.0	0.0	-0.826374	0.180396	0.431328	1.597404
32845	2017-09-30 20:00:00	0.322137	1.0	0.0	-0.826374	0.111436	0.003354	1.741992
32846	2017-09-30 21:00:00	0.251104	1.0	0.0	-0.826374	0.042475	-0.396087	1.886580
32847	2017-09-30 22:00:00	0.241324	1.0	0.0	-0.826374	0.111436	-0.595808	2.031169
32848	2017-09-30 23:00:00	0.271460	1.0	0.0	-0.826374	0.180396	-0.824060	2.175757

32849 rows × 8 columns

In [62]:

```
# Scatter plot between % Baseline and sunHour
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
plt.scatter(analysis_df['sunHour'], analysis_df['% Baseline'], alpha=0.5)
plt.title('% Baseline vs SunHour')
plt.xlabel('SunHour')
plt.ylabel('% Baseline')

# Scatter plot between % Baseline and humidity
plt.subplot(1, 2, 2)
```

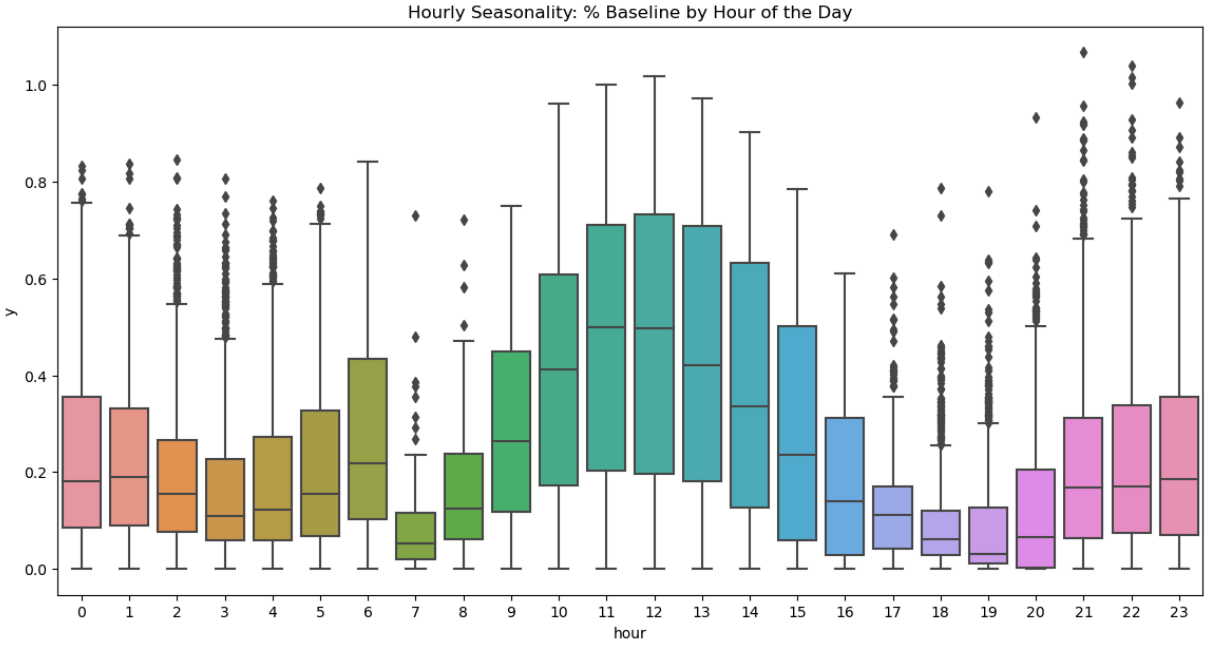
```
plt.scatter(analysis_df['humidity'], analysis_df['% Baseline'], alpha=0.5)
plt.title('% Baseline vs Humidity')
plt.xlabel('Humidity')
plt.ylabel('% Baseline')

plt.tight_layout()
plt.show()
```

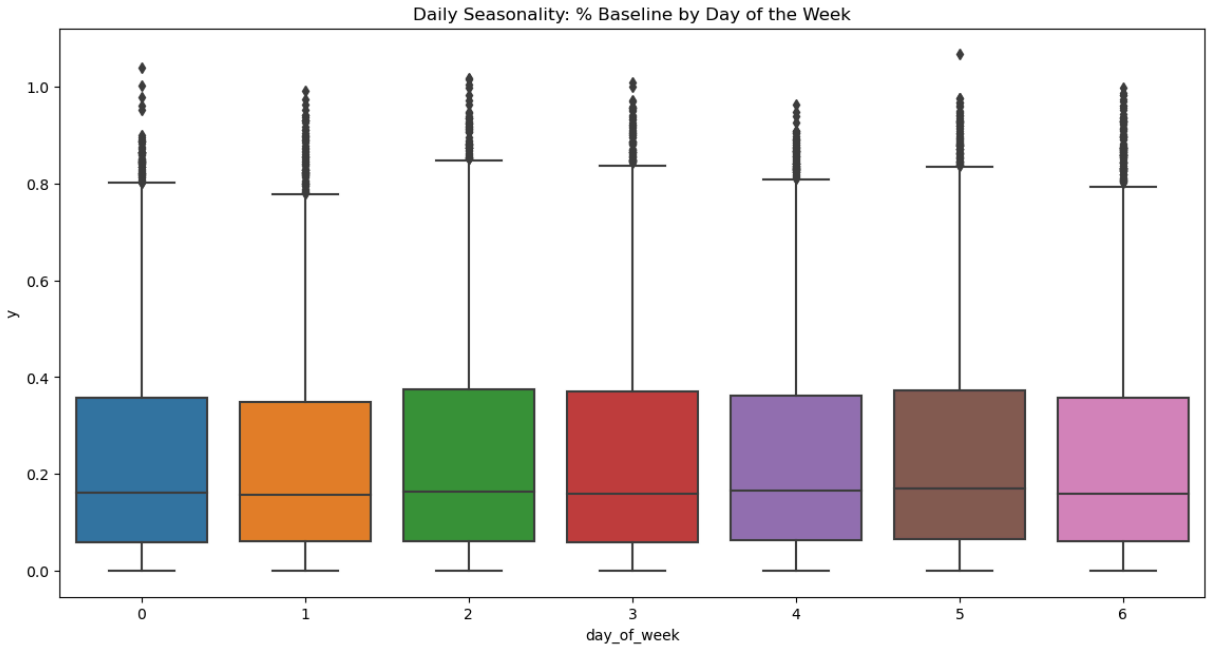


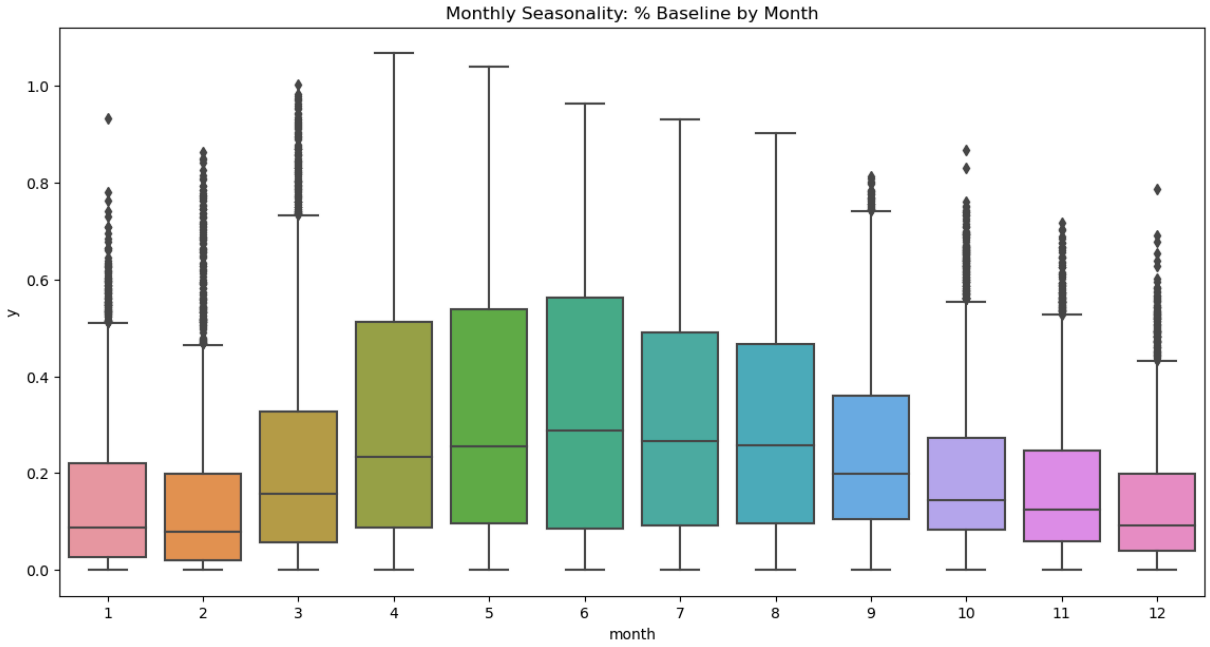
In [63]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Assume 'train_df' is your DataFrame and 'Timestamp' is your time column
#train_df['Timestamp'] = pd.to_datetime(train_df['Timestamp'])

# Extract time-based features
train_df['hour'] = train_df['ds'].dt.hour
train_df['day_of_week'] = train_df['ds'].dt.dayofweek
train_df['month'] = train_df['ds'].dt.month

# Plot boxplot for hourly seasonality
plt.figure(figsize=(14, 7))
sns.boxplot(x='hour', y='y', data=train_df)
plt.title('Hourly Seasonality: % Baseline by Hour of the Day')
plt.show()

# Plot boxplot for daily seasonality
plt.figure(figsize=(14, 7))
sns.boxplot(x='day_of_week', y='y', data=train_df)
plt.title('Daily Seasonality: % Baseline by Day of the Week')
plt.show()

# Plot boxplot for monthly seasonality
plt.figure(figsize=(14, 7))
sns.boxplot(x='month', y='y', data=train_df)
plt.title('Monthly Seasonality: % Baseline by Month')
plt.show()
```

```
C:\Users\NAUFAL\AppData\Local\Temp\ipykernel_28172\1788810459.py:9: SettingWithCopyW
arning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  train_df['hour'] = train_df['ds'].dt.hour
C:\Users\NAUFAL\AppData\Local\Temp\ipykernel_28172\1788810459.py:10: SettingWithCopy
Warning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  train_df['day_of_week'] = train_df['ds'].dt.dayofweek
C:\Users\NAUFAL\AppData\Local\Temp\ipykernel_28172\1788810459.py:11: SettingWithCopy
Warning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  train_df['month'] = train_df['ds'].dt.month
```

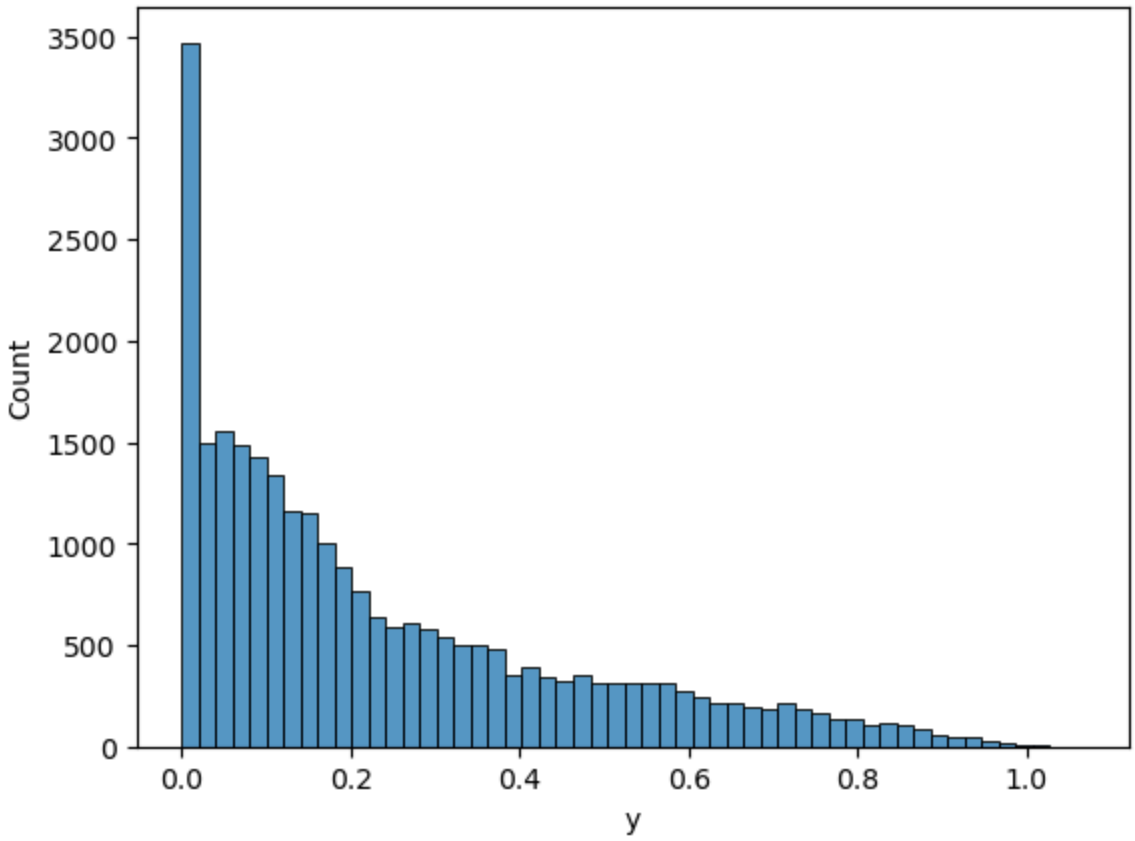






In [64]:
```
sns.histplot(train_df['y'])
plt.show
```

Out[64]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

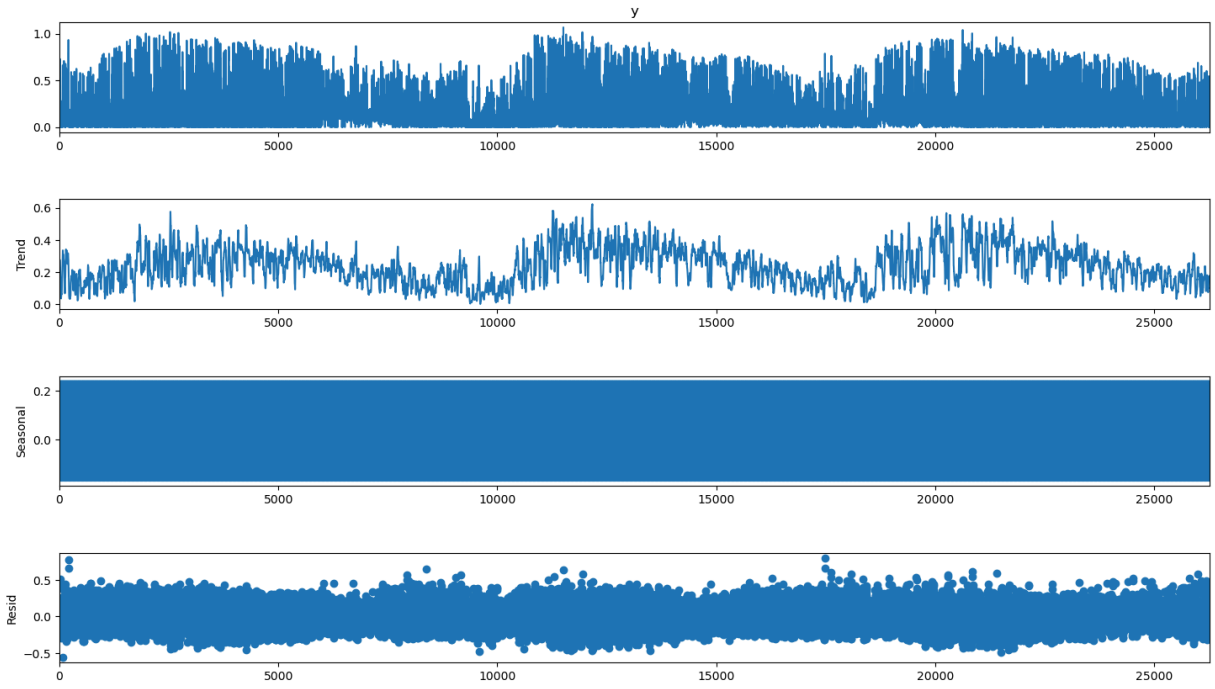


In [65]:
```
# Calculate Q1 (25th percentile) and Q3 (75th percentile)
Q1 = train_df['y'].quantile(0.25)
Q3 = train_df['y'].quantile(0.75)
IQR = Q3 - Q1

# Define outliers as values below Q1 - 1.5*IQR or above Q3 + 1.5*IQR
tol = 2
outliers = train_df[(train_df['y'] < (Q1 - tol * IQR)) | (train_df['y'] > (Q3 + tol
print(f"Number of outliers detected: {len(outliers)}")
```

```
Number of outliers detected: 25
```

In [66]:
```
result = seasonal_decompose(train_df['y'], model='additive',period=24)
fig = plt.figure()
fig = result.plot()
fig.set_size_inches(16, 9)
```

```
<Figure size 640x480 with 0 Axes>
```



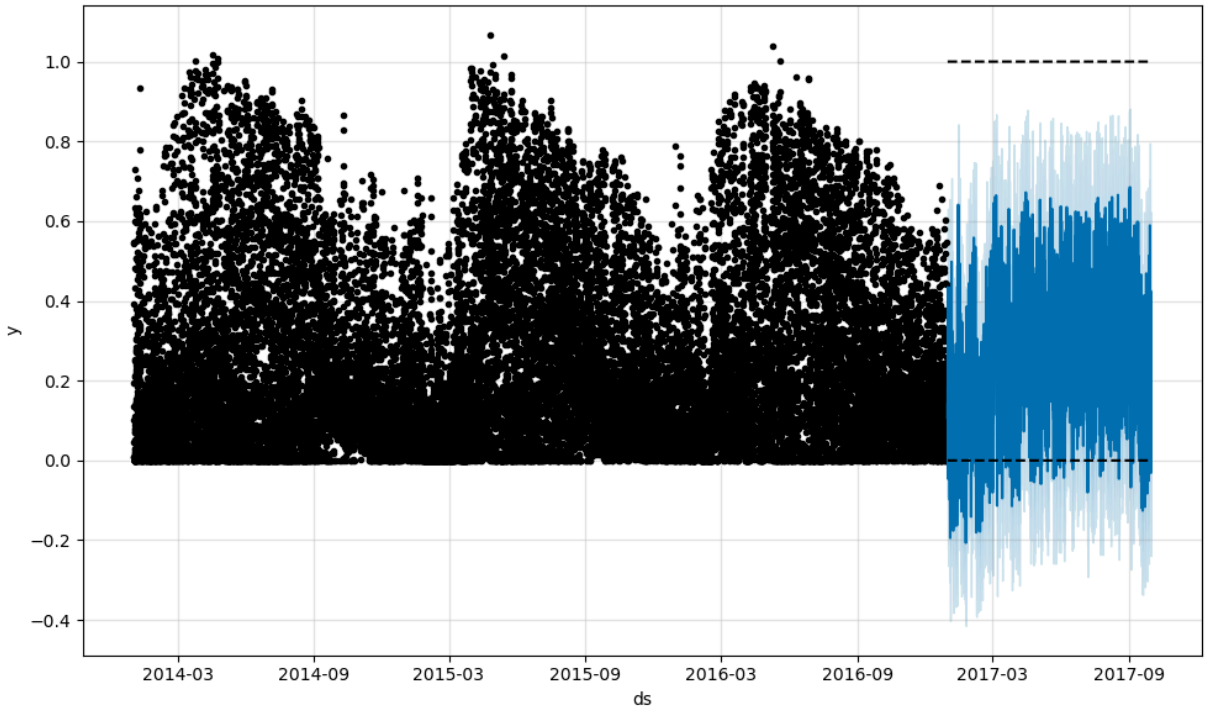
In [67]:
```
# Initialize the Prophet model
model = Prophet(
    growth='logistic',  # Set to 'logistic' if using cap and floor
    changepoint_prior_scale=0.75,  # Adjust this for more/less trend flexibility
    seasonality_prior_scale=10  # Control the flexibility of seasonality
)

# Add custom hourly seasonality
model.add_seasonality(name='hourly', period=24, fourier_order=15)
# Add monthly seasonality
model.add_seasonality(name='monthly', period=30.5, fourier_order=15)

# Add humidity and sunHour as regressors
model.add_regressor('shumidity')
model.add_regressor('ssunHour')
model.add_regressor('scloudcover')
model.add_regressor('shour_month')

# Fit the model to the data
model.fit(train_df)
```

```
22:57:17 - cmdstanpy - INFO - Chain [1] start processing
22:57:31 - cmdstanpy - INFO - Chain [1] done processing
```

Out[67]: <prophet.forecaster.Prophet at 0x264107f71c0>

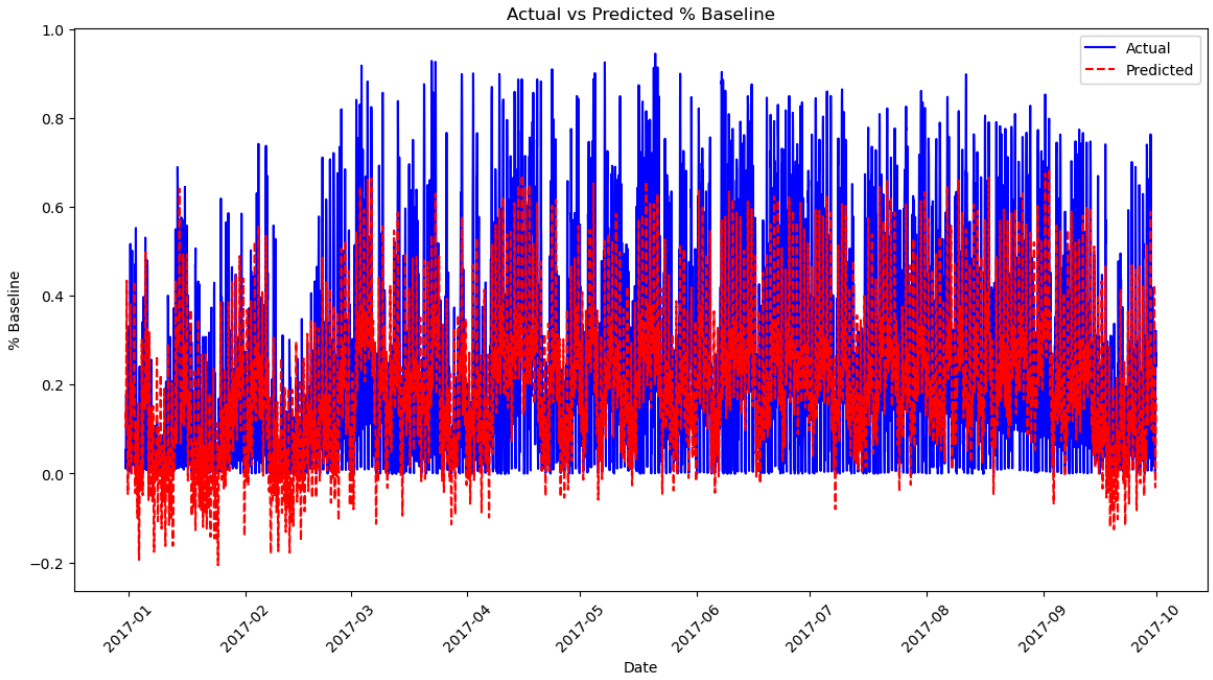
In [68]:
```
# Use the trained Prophet model to make predictions
future_test = test_df.copy()
forecast = model.predict(future_test)

# Merge the forecast with the test set (to include actual values for comparison)
test_predictions = pd.merge(test_df, forecast[['ds', 'yhat']], on='ds', how='left')
```

In [69]:
```
# Plot the forecast
model.plot(forecast)
plt.show()
```



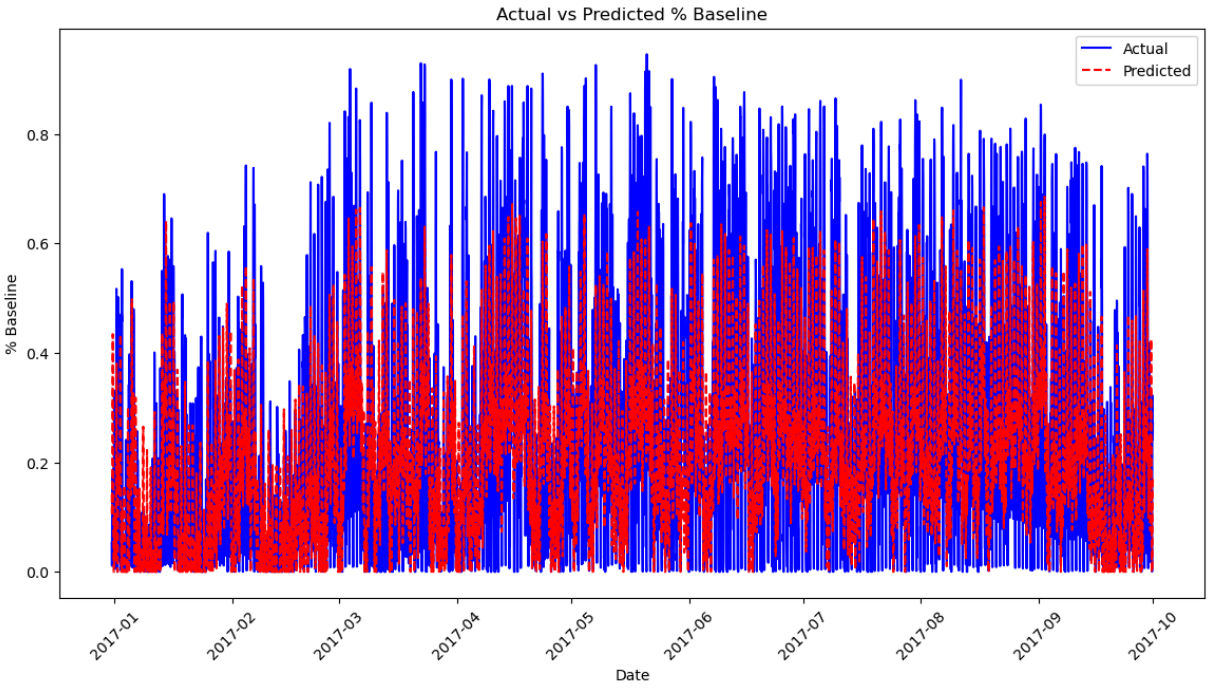
In [70]:
```
def rmse(x,y):
    return np.sqrt(mean_squared_error(x,y))
```

In [71]:
```
# Assuming 'test_predictions' contains 'ds' (timestamps), 'y' (actual values), and
plt.figure(figsize=(14, 7))

# Plot the actual values
plt.plot(test_predictions['ds'], test_predictions['y'], label='Actual', color='blue

# Plot the predicted values
plt.plot(test_predictions['ds'], test_predictions['yhat'], label='Predicted', color

# Add labels and title
plt.xlabel('Date')
plt.ylabel('% Baseline')
plt.title('Actual vs Predicted % Baseline')
plt.legend()

# Rotate the x-axis labels for better readability
plt.xticks(rotation=45)

# Display the plot
plt.show()
```



In [72]:

```
# Assuming 'forecast' is your DataFrame with the predicted values
test_predictions['yhat1'] = test_predictions['yhat'].apply(lambda x: max(x, 0))
# Assuming 'test_predictions' contains 'ds' (timestamps), 'y' (actual values), and
plt.figure(figsize=(14, 7))

# Plot the actual values
plt.plot(test_predictions['ds'], test_predictions['y'], label='Actual', color='blue

# Plot the predicted values
plt.plot(test_predictions['ds'], test_predictions['yhat1'], label='Predicted', colo

# Add labels and title
plt.xlabel('Date')
plt.ylabel('% Baseline')
plt.title('Actual vs Predicted % Baseline')
plt.legend()

# Rotate the x-axis labels for better readability
plt.xticks(rotation=45)

# Display the plot
plt.show()
```



In [73]:
```
print(rmse(test_predictions['yhat1'],test_predictions['y']))
print(rmse(test_predictions['yhat'],test_predictions['y']))
```

```
0.17355783250325976
0.17648716554008712
```

PRED

In [77]:
```
# Rename the 'Timestamp' column in ttest to 'ds'
#ttest.rename(columns={'Timestamp': 'ds'}, inplace=True)
ttest[['shumidity', 'ssunHour','scloudcover','shour_month']] = scaler.fit_transform
ttest
```

```
C:\Users\NAUFAL\AppData\Local\Temp\ipykernel_28172\3652278203.py:3: SettingWithCopyW
arning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  ttest[['shumidity', 'ssunHour','scloudcover','shour_month']] = scaler.fit_transfor
m(ttest[['humidity', 'sunHour','cloudcover','hour_month']])
C:\Users\NAUFAL\AppData\Local\Temp\ipykernel_28172\3652278203.py:3: SettingWithCopyW
arning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  ttest[['shumidity', 'ssunHour','scloudcover','shour_month']] = scaler.fit_transfor
m(ttest[['humidity', 'sunHour','cloudcover','hour_month']])
C:\Users\NAUFAL\AppData\Local\Temp\ipykernel_28172\3652278203.py:3: SettingWithCopyW
arning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  ttest[['shumidity', 'ssunHour','scloudcover','shour_month']] = scaler.fit_transfor
m(ttest[['humidity', 'sunHour','cloudcover','hour_month']])
C:\Users\NAUFAL\AppData\Local\Temp\ipykernel_28172\3652278203.py:3: SettingWithCopyW
arning:
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/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  ttest[['shumidity', 'ssunHour','scloudcover','shour_month']] = scaler.fit_transfor
m(ttest[['humidity', 'sunHour','cloudcover','hour_month']])
```

Out[77]:

	ds	% Baseline	maxtempC	mintempC	sunHour	uvIndex	DewPointC	FeelsLike
32849	2017-10-01 00:00:00	0.237560	21	9	11.6	5	7	
32850	2017-10-01 01:00:00	0.352637	21	9	11.6	5	6	
32851	2017-10-01 02:00:00	0.192091	21	9	11.6	5	6	
32852	2017-10-01 03:00:00	0.118935	21	9	11.6	5	6	
32853	2017-10-01 04:00:00	0.119415	21	9	11.6	5	7	
...	...	...	...	...	...	...	...	
35052	2017-12-31 19:00:00	0.123297	-9	-13	8.7	1	-17	-2
35053	2017-12-31 20:00:00	0.147535	-9	-13	8.7	1	-17	-2
35054	2017-12-31 21:00:00	0.180168	-9	-13	8.7	1	-17	-2
35055	2017-12-31 22:00:00	0.248257	-9	-13	8.7	1	-17	-2
35056	2017-12-31 23:00:00	0.216462	-9	-13	8.7	1	-17	-2

2208 rows × 32 columns

In [80]:
```
prophet_df
```

Out[80]:

	ds	y	cap	floor	ssunHour	shumidity	scloudcover	shour_month
0	2014-01-01 07:00:00	0.007900	1.0	0.0	-0.422555	-1.681535	-0.795529	-1.037312
1	2014-01-01 08:00:00	0.101900	1.0	0.0	-0.422555	-1.888416	-0.852592	-1.021246
2	2014-01-01 09:00:00	0.393200	1.0	0.0	-0.422555	-2.095297	-0.909655	-1.005181
3	2014-01-01 10:00:00	0.544700	1.0	0.0	-0.422555	-2.095297	-0.738466	-0.989116
4	2014-01-01 11:00:00	0.548500	1.0	0.0	-0.422555	-2.164257	-0.567277	-0.973050
...	...	...	...	...	...	...	...	...
32844	2017-09-30 19:00:00	0.189902	1.0	0.0	-0.826374	0.180396	0.431328	1.597404
32845	2017-09-30 20:00:00	0.322137	1.0	0.0	-0.826374	0.111436	0.003354	1.741992
32846	2017-09-30 21:00:00	0.251104	1.0	0.0	-0.826374	0.042475	-0.396087	1.886580
32847	2017-09-30 22:00:00	0.241324	1.0	0.0	-0.826374	0.111436	-0.595808	2.031169
32848	2017-09-30 23:00:00	0.271460	1.0	0.0	-0.826374	0.180396	-0.824060	2.175757

32849 rows × 8 columns

In [79]:

```
# Create a date range from 1st October 2014 00:00 to 31st December 2017 23:00
date_range = pd.date_range(start='2017-10-01 00:00', end='2017-12-31 23:00', freq='
# Create a DataFrame with this date range
date_df = pd.DataFrame(date_range, columns=['ds'])
date_df = pd.merge(date_df, ttest[['ds', 'shumidity', 'ssunHour','scloudcover','sho
date_df
```

Out[79]:

	ds	shumidity	ssunHour	scloudcover	shour_month
0	2017-10-01 00:00:00	0.644684	1.666754	-0.983866	-1.644198
1	2017-10-01 01:00:00	0.713048	1.666754	-1.011822	-1.514222
2	2017-10-01 02:00:00	0.781411	1.666754	-1.067734	-1.384246
3	2017-10-01 03:00:00	0.781411	1.666754	-1.095690	-1.254270
4	2017-10-01 04:00:00	0.713048	1.666754	-1.095690	-1.124294
...	...	...	...	...	...
2203	2017-12-31 19:00:00	-0.107313	0.471993	-0.983866	1.319258
2204	2017-12-31 20:00:00	-0.038950	0.471993	-1.039778	1.475229
2205	2017-12-31 21:00:00	0.029414	0.471993	-1.095690	1.631200
2206	2017-12-31 22:00:00	0.097777	0.471993	-1.095690	1.787172
2207	2017-12-31 23:00:00	0.166140	0.471993	-1.095690	1.943143

2208 rows × 5 columns

In [83]:
```
# Create a date range from 1st October 2014 00:00 to 31st December 2017 23:00
date_range = pd.date_range(start='2017-10-01 00:00', end='2017-12-31 23:00', freq='
# Create a DataFrame with this date range
date_df = pd.DataFrame(date_range, columns=['ds'])
# Ensure the future dataframe also has cap and floor values
date_df['cap'] = 1.0  # Same cap as in the training data
date_df['floor'] = 0  # Same floor as in the training data
date_df = pd.merge(date_df, ttest[['ds', 'shumidity', 'ssunHour','scloudcover','sho
# Reorder the columns to swap 'ssunHour' and 'shumidity'
date_df = date_df[['ds', 'cap', 'floor', 'ssunHour', 'shumidity','scloudcover','sho
forecast = model.predict(date_df)
# Merge the forecast with the test set (to include actual values for comparison)
pred = pd.merge(date_df, forecast[['ds', 'yhat']], on='ds', how='left')
```

In [84]:
```
submit = pd.read_csv('dataset/sample_submission.csv')
# Convert 'Timestamp' in submit to datetime format
a = submit.copy()
submit['Timestamp'] = pd.to_datetime(submit['Timestamp'], format='%b %d, %Y %I%p')

# Filter the pred DataFrame to include only the rows with matching timestamps in su
filtered_pred = pred[pred['ds'].isin(submit['Timestamp'])]
filtered_pred
```

Out[84]:

	ds	cap	floor	ssunHour	shumidity	scloudcover	shour_month	yhat
6	2017-10-01 06:00:00	1.0	0	1.666754	0.507958	-1.095690	-0.864341	0.117762
7	2017-10-01 07:00:00	1.0	0	1.666754	-0.107313	-1.095690	-0.734365	0.125560
8	2017-10-01 08:00:00	1.0	0	1.666754	-0.654221	-1.095690	-0.604389	0.172740
9	2017-10-01 09:00:00	1.0	0	1.666754	-1.201128	-1.095690	-0.474413	0.284134
10	2017-10-01 10:00:00	1.0	0	1.666754	-1.474582	-1.095690	-0.344437	0.410183
...	...	...	...	...	...	...	...	...
2196	2017-12-31 12:00:00	1.0	0	0.471993	-0.927674	-0.480658	0.227458	0.398960
2197	2017-12-31 13:00:00	1.0	0	0.471993	-0.790947	-0.284966	0.383430	0.339246
2198	2017-12-31 14:00:00	1.0	0	0.471993	-0.722584	-0.061318	0.539401	0.247683
2199	2017-12-31 15:00:00	1.0	0	0.471993	-0.585857	0.134374	0.695372	0.150915
2200	2017-12-31 16:00:00	1.0	0	0.471993	-0.449130	-0.201098	0.851344	0.079800

1077 rows × 8 columns

In [85]:

```
# Reset the index of both DataFrames before concatenation
a_reset = a.reset_index(drop=True)
filtered_pred_reset = filtered_pred.reset_index(drop=True)

# Concatenate the 'Timestamp' from a and 'yhat' from filtered_pred
result_df = pd.concat([a_reset['Timestamp'], filtered_pred_reset['yhat']], axis=1)

# Rename the columns if needed
result_df.columns = a.columns
result_df
```

Out[85]:

	Timestamp	% Baseline
0	Oct 1, 2017 6am	0.117762
1	Oct 1, 2017 7am	0.125560
2	Oct 1, 2017 8am	0.172740
3	Oct 1, 2017 9am	0.284134
4	Oct 1, 2017 10am	0.410183
...	...	...
1072	Dec 31, 2017 12pm	0.398960
1073	Dec 31, 2017 1pm	0.339246
1074	Dec 31, 2017 2pm	0.247683
1075	Dec 31, 2017 3pm	0.150915
1076	Dec 31, 2017 4pm	0.079800

1077 rows × 2 columns

In [86]:
```
result_df.to_csv('submit/submission_prophetv3.csv',index=False)
```

In [ ]:
```
```