FourCastNet Evaluation using ERA5.

Loading Datasets

```
import xarray as xr
ds_fc = xr.open_dataset("fourcastnetv2-small_20210601_0000.grib2.nc")
```

```
import xarray as xr
ds_fc2 = "/content/drive/MyDrive/Fourcastnet/forecastnet_input_final.nc"
ds1 = xr.open_dataset(ds_fc2)
```

Subsetting Datasets to India Region

```
# Define India region
ds_fc_india = ds_fc.sel(latitude=slice(35, 5), longitude=slice(65, 100))
ds1_india = ds1.sel(latitude=slice(35, 5), longitude=slice(65, 100))
```

```
# Interpolate ERA5 to ForecastNet times
msl_era5_interp = ds1_india["msl"].interp(valid_time=ds_fc_india.time)
```

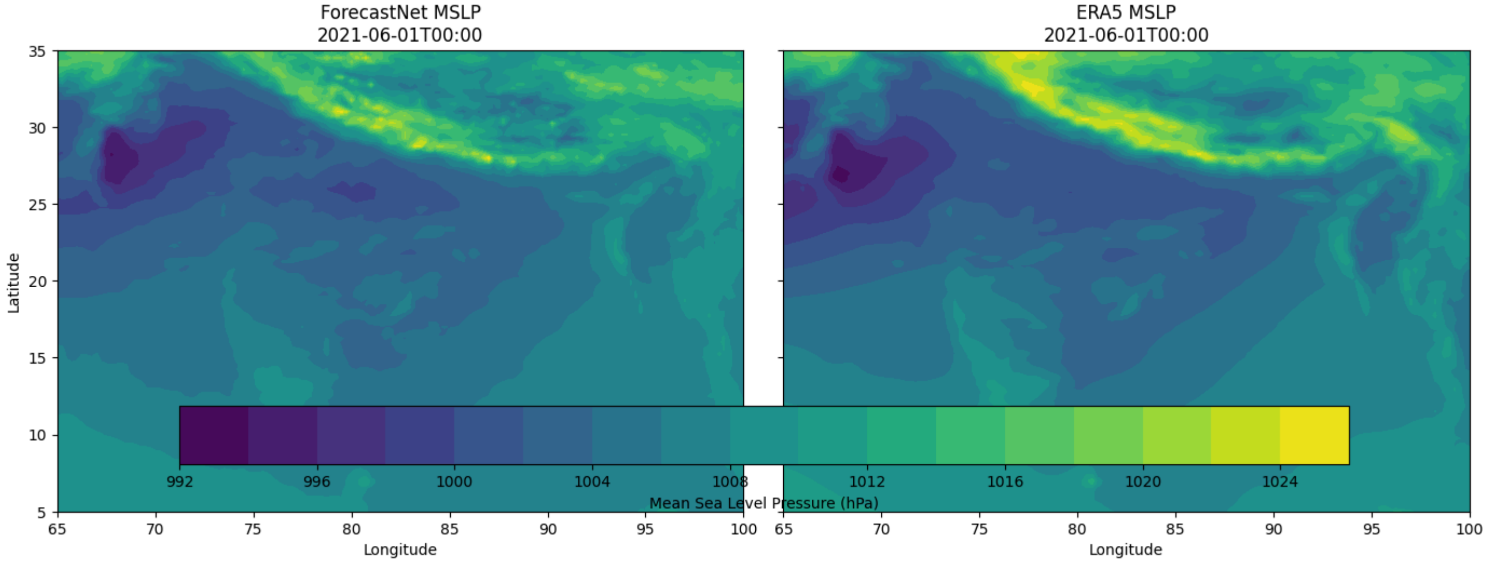
```
# Drop pressure_level if it exists
if "pressure_level" in msl_era5_interp.dims:
    msl_era5_interp = msl_era5_interp.isel(pressure_level=0, drop=True)
```

Variable : Mean Sea Level Pressure (mslp)

```
import matplotlib.pyplot as plt
import ipywidgets as widgets
from ipywidgets import interact
import numpy as np

# Convert to hPa
msl_fc_hpa = ds_fc_india["msl"] / 100
msl_era5_hpa = msl_era5_interp / 100

# Drop pressure level if mistakenly included
if "pressure_level" in msl_era5_hpa.dims:
    msl_era5_hpa = msl_era5_hpa.isel(pressure_level=0, drop=True)

# Create plotting function
def plot_mslp_interactive(time_idx):
    fig, axs = plt.subplots(1, 2, figsize=(14, 5), sharex=True, sharey=True)

    # ForecastNet
    cf1 = axs[0].contourf(ds_fc_india.longitude, ds_fc_india.latitude,
                          msl_fc_hpa.isel(time=time_idx), levels=20, cmap="viridis")
    axs[0].set_title(f"ForecastNet MSLP\n{str(ds_fc_india.time.values[time_idx])[:16]}")
    axs[0].set_xlabel("Longitude")
    axs[0].set_ylabel("Latitude")

    # ERA5
    cf2 = axs[1].contourf(ds1_india.longitude, ds1_india.latitude,
                          msl_era5_hpa.isel(time=time_idx), levels=20, cmap="viridis")
    axs[1].set_title(f"ERA5 MSLP\n{str(ds_fc_india.time.values[time_idx])[:16]}")
    axs[1].set_xlabel("Longitude")

    cbar = fig.colorbar(cf1, ax=axs, orientation='horizontal', pad=0.08)
    cbar.set_label("Mean Sea Level Pressure (hPa)")

    plt.tight_layout()
    plt.show()

# Create interactive slider
interact(plot_mslp_interactive, time_idx=widgets.IntSlider(min=0, max=len(ds_fc_india.time)-1, step=1, value=0));
```

time_idx ○ 0

```
/tmp/ipython-input-10-2023116686.py:34: UserWarning: This figure includes Axes that are not compatible with tight_layout, so results might be
incorrect.
  plt.tight_layout()
```



ForecastNet MSLP
2021-06-01T00:00

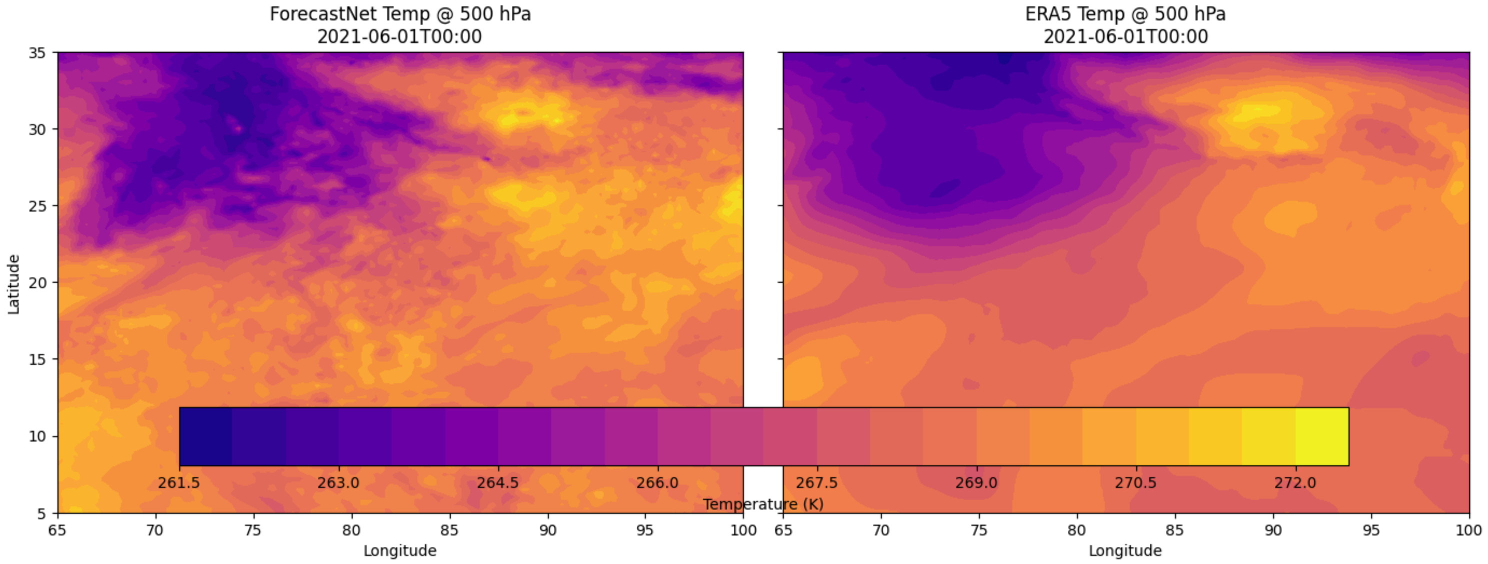
ERA5 MSLP
2021-06-01T00:00

Variable : Temperature (t)

```
import matplotlib.pyplot as plt
import ipywidgets as widgets
from ipywidgets import interact

# Choose pressure level to visualize
level = 500

# Select temperature at 500 hPa
t_fc_500 = ds_fc_india["t"].sel(level=level)
t_era5_500 = ds1_india["t"].sel(pressure_level=level)

# Interpolate ERA5 time to match ForecastNet
t_era5_interp = t_era5_500.interp(valid_time=ds_fc_india.time)

# Interactive plotting function
def plot_temp_interactive(time_idx):
    fig, axs = plt.subplots(1, 2, figsize=(14, 5), sharex=True, sharey=True)

    # ForecastNet temp
    cf1 = axs[0].contourf(ds_fc_india.longitude, ds_fc_india.latitude,
                          t_fc_500.isel(time=time_idx), levels=20, cmap="plasma")
    axs[0].set_title(f"ForecastNet Temp @ {level} hPa\n{str(ds_fc_india.time.values[time_idx])[:16]}")
    axs[0].set_xlabel("Longitude")
    axs[0].set_ylabel("Latitude")

    # ERA5 temp
    cf2 = axs[1].contourf(ds1_india.longitude, ds1_india.latitude,
                          t_era5_interp.isel(time=time_idx), levels=20, cmap="plasma")
    axs[1].set_title(f"ERA5 Temp @ {level} hPa\n{str(ds_fc_india.time.values[time_idx])[:16]}")
    axs[1].set_xlabel("Longitude")

    # Shared colorbar
    cbar = fig.colorbar(cf1, ax=axs, orientation='horizontal', pad=0.08)
    cbar.set_label("Temperature (K)")

    plt.tight_layout()
    plt.show()

# Interactive slider
interact(plot_temp_interactive, time_idx=widgets.IntSlider(min=0, max=len(ds_fc_india.time)-1, step=1, value=0));
```

time_idx 0

```
/tmp/ipython-input-23-3397455138.py:36: UserWarning: This figure includes Axes that are not compatible with tight_layout, so results might be
incorrect.
  plt.tight_layout()
```

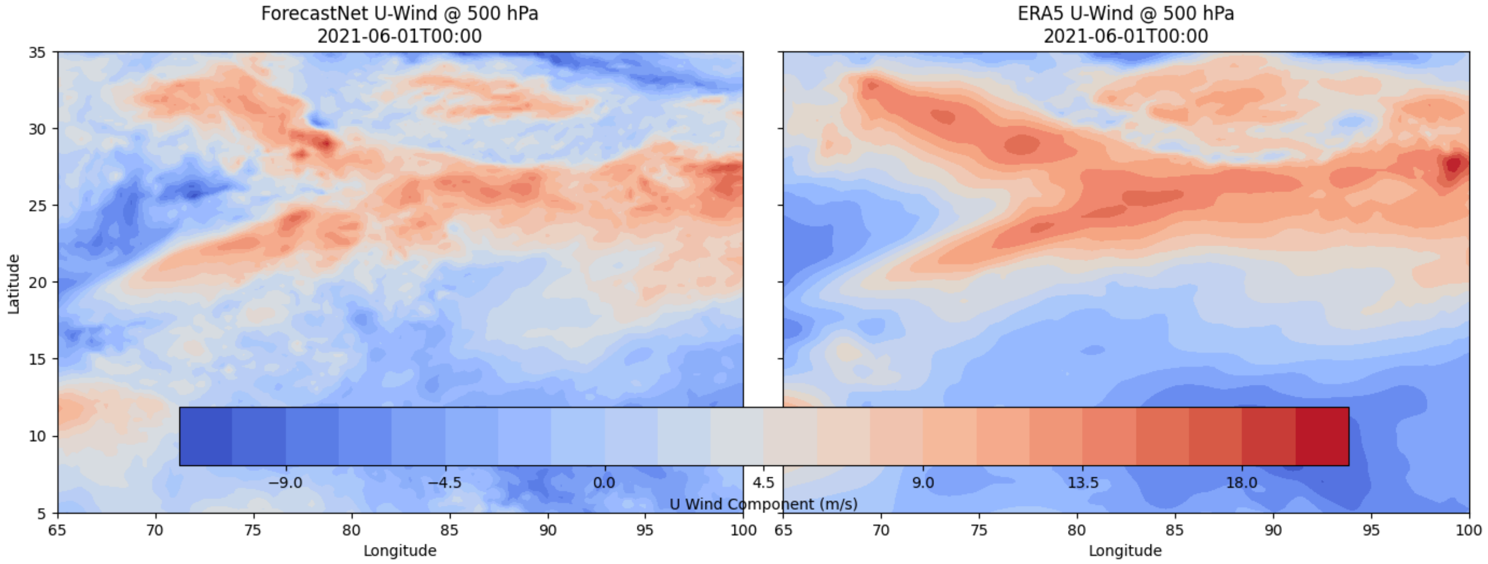


Variable : U-Wind

```
import matplotlib.pyplot as plt
import ipywidgets as widgets
from ipywidgets import interact

# Select pressure level
level = 500

# Slice U wind at 500 hPa
u_fc_500 = ds_fc_india["u"].sel(level=level)
u_era5_500 = ds1_india["u"].sel(pressure_level=level)

# Interpolate ERA5 U-wind to ForecastNet time steps
u_era5_interp = u_era5_500.interp(valid_time=ds_fc_india.time)

# Define interactive plotting function
def plot_u_interactive(time_idx):
    fig, axs = plt.subplots(1, 2, figsize=(14, 5), sharex=True, sharey=True)

    # ForecastNet U-wind
    cf1 = axs[0].contourf(ds_fc_india.longitude, ds_fc_india.latitude,
                          u_fc_500.isel(time=time_idx), levels=20, cmap="coolwarm")
    axs[0].set_title(f"ForecastNet U-Wind @ {level} hPa\n{str(ds_fc_india.time.values[time_idx])[:16]}")
    axs[0].set_xlabel("Longitude")
    axs[0].set_ylabel("Latitude")

    # ERA5 U-wind
    cf2 = axs[1].contourf(ds1_india.longitude, ds1_india.latitude,
                          u_era5_interp.isel(time=time_idx), levels=20, cmap="coolwarm")
    axs[1].set_title(f"ERA5 U-Wind @ {level} hPa\n{str(ds_fc_india.time.values[time_idx])[:16]}")
    axs[1].set_xlabel("Longitude")

    # Shared colorbar
    cbar = fig.colorbar(cf1, ax=axs, orientation='horizontal', pad=0.08)
    cbar.set_label("U Wind Component (m/s)")

    plt.tight_layout()
    plt.show()

# Interactive slider
interact(plot_u_interactive, time_idx=widgets.IntSlider(min=0, max=len(ds_fc_india.time)-1, step=1, value=0));
```

time_idx 0

```
/tmp/ipython-input-12-2440011192.py:36: UserWarning: This figure includes Axes that are not compatible with tight_layout, so results might be incorrect.
  plt.tight_layout()
```

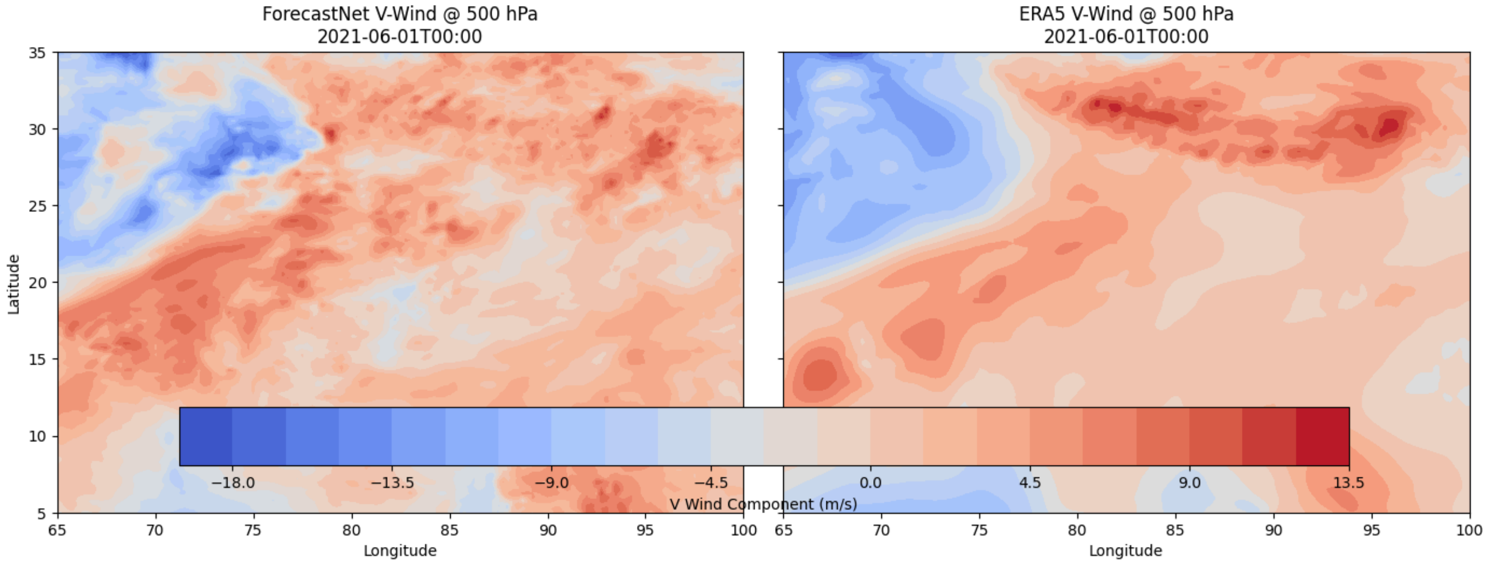


Variable : V-Wind

```
import matplotlib.pyplot as plt
import ipywidgets as widgets
from ipywidgets import interact

# Define pressure level
level = 500

# Slice the V wind data
v_fc_500 = ds_fc_india["v"].sel(level=level)
v_era5_500 = ds1_india["v"].sel(pressure_level=level)

# Interpolate ERA5 V wind to ForecastNet time
v_era5_interp = v_era5_500.interp(valid_time=ds_fc_india.time)

# Define plotting function
def plot_v_interactive(time_idx):
    fig, axs = plt.subplots(1, 2, figsize=(14, 5), sharex=True, sharey=True)

    # ForecastNet V wind
    cf1 = axs[0].contourf(ds_fc_india.longitude, ds_fc_india.latitude,
                          v_fc_500.isel(time=time_idx), levels=20, cmap="coolwarm")
    axs[0].set_title(f"ForecastNet V-Wind @ {level} hPa\n{str(ds_fc_india.time.values[time_idx])[:16]}")
    axs[0].set_xlabel("Longitude")
    axs[0].set_ylabel("Latitude")

    # ERA5 V wind
    cf2 = axs[1].contourf(ds1_india.longitude, ds1_india.latitude,
                          v_era5_interp.isel(time=time_idx), levels=20, cmap="coolwarm")
    axs[1].set_title(f"ERA5 V-Wind @ {level} hPa\n{str(ds_fc_india.time.values[time_idx])[:16]}")
    axs[1].set_xlabel("Longitude")

    # Shared colorbar
    cbar = fig.colorbar(cf1, ax=axs, orientation='horizontal', pad=0.08)
    cbar.set_label("V Wind Component (m/s)")

    plt.tight_layout()
    plt.show()

# Create interactive slider
interact(plot_v_interactive, time_idx=widgets.IntSlider(min=0, max=len(ds_fc_india.time)-1, step=1, value=0));
```

time_idx ○ 0

```
/tmp/ipython-input-13-3518229210.py:36: UserWarning: This figure includes Axes that are not compatible with tight_layout, so results might be incorrect.
  plt.tight_layout()
```



ForecastNet V-Wind @ 500 hPa
2021-06-01T00:00

ERA5 V-Wind @ 500 hPa
2021-06-01T00:00

RMSE Results

```
rmse_df
```

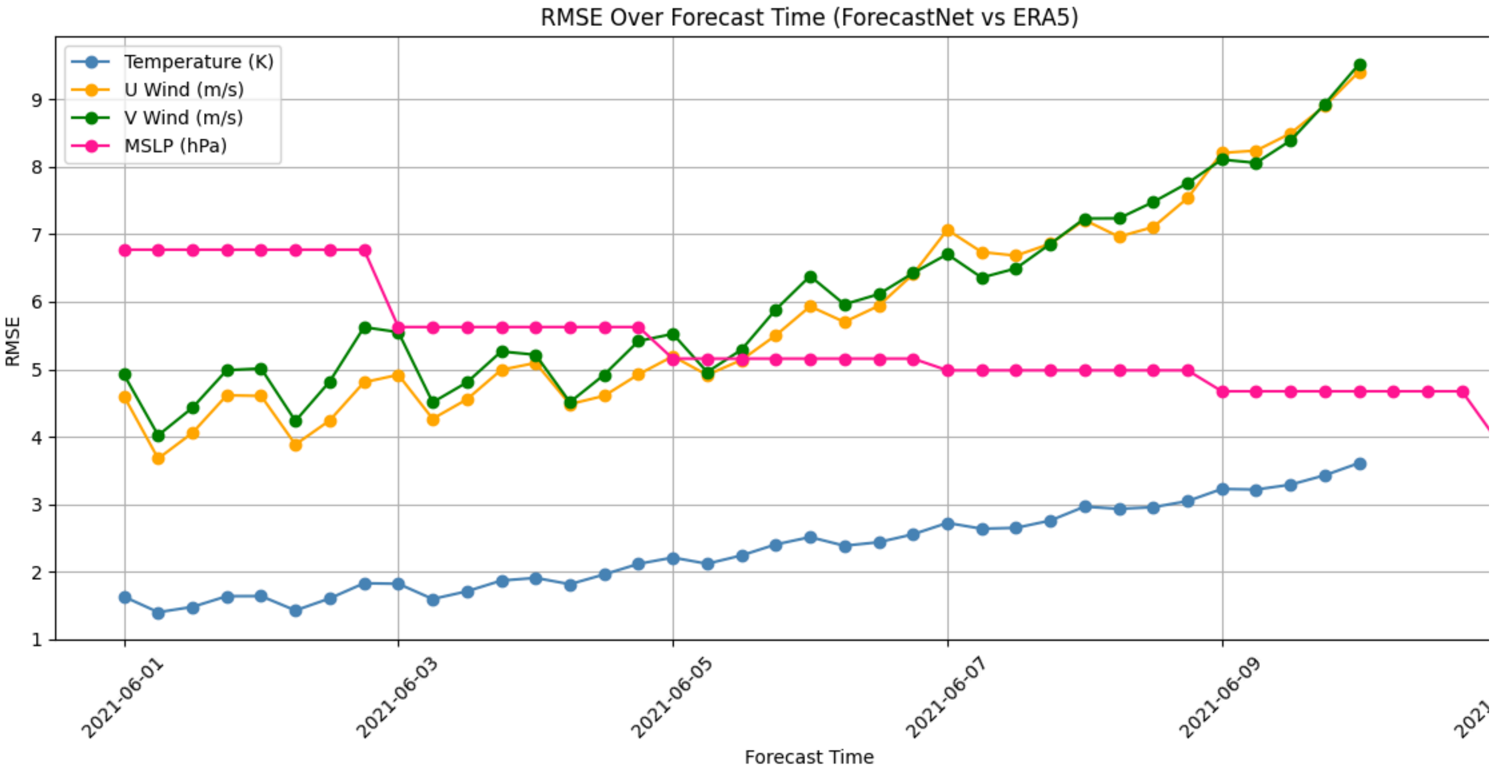
Forecast Time	Temperature (K)	U Wind (m/s)	V Wind (m/s)	MSLP (hPa)
2021-06-01 00:00:00	1.633047	4.604428	4.923812	6.770746
2021-06-01 06:00:00	1.401908	3.676972	4.021385	6.770746
2021-06-01 12:00:00	1.480447	4.058488	4.430555	6.770746
2021-06-01 18:00:00	1.639101	4.614280	4.986223	6.770746
2021-06-02 00:00:00	1.640600	4.605251	5.011039	6.770746
2021-06-02 06:00:00	1.428128	3.885436	4.234517	6.770746
2021-06-02 12:00:00	1.606495	4.242083	4.818582	6.770746
2021-06-02 18:00:00	1.830806	4.807811	5.623042	6.770746
2021-06-03 00:00:00	1.822931	4.917695	5.549398	5.625687
2021-06-03 06:00:00	1.597293	4.266328	4.510089	5.625687
2021-06-03 12:00:00	1.711666	4.555160	4.805961	5.625687
2021-06-03 18:00:00	1.870107	4.991476	5.263682	5.625687
2021-06-04 00:00:00	1.910250	5.093464	5.214086	5.625687
2021-06-04 06:00:00	1.816893	4.483365	4.512112	5.625687
2021-06-04 12:00:00	1.961250	4.606370	4.919582	5.625687
2021-06-04 18:00:00	2.118309	4.924431	5.418070	5.625687
2021-06-05 00:00:00	2.211281	5.192384	5.523280	5.157331
2021-06-05 06:00:00	2.119633	4.915357	4.958220	5.157331
2021-06-05 12:00:00	2.242859	5.137439	5.289418	5.157331
2021-06-05 18:00:00	2.405692	5.502378	5.884184	5.157331
2021-06-06 00:00:00	2.514188	5.931533	6.380753	5.157331
2021-06-06 06:00:00	2.386177	5.699921	5.961348	5.157331
2021-06-06 12:00:00	2.438905	5.939512	6.112399	5.157331
2021-06-06 18:00:00	2.555645	6.407555	6.423912	5.157331
2021-06-07 00:00:00	2.724514	7.065458	6.704053	4.983936
2021-06-07 06:00:00	2.638366	6.735491	6.357178	4.983936
2021-06-07 12:00:00	2.651054	6.682758	6.493703	4.983936
2021-06-07 18:00:00	2.759269	6.858896	6.856412	4.983936
2021-06-08 00:00:00	2.966004	7.209506	7.232868	4.983936
2021-06-08 06:00:00	2.932446	6.962589	7.234990	4.983936
2021-06-08 12:00:00	2.957193	7.106282	7.476828	4.983936
2021-06-08 18:00:00	3.047305	7.534572	7.756034	4.983936
2021-06-09 00:00:00	3.227284	8.203882	8.105322	4.673746
2021-06-09 06:00:00	3.218655	8.238231	8.056571	4.673746
2021-06-09 12:00:00	3.290741	8.492893	8.386154	4.673746
2021-06-09 18:00:00	3.429882	8.900175	8.927583	4.673746
2021-06-10 00:00:00	3.611019	9.407978	9.519715	4.673746
2021-06-10 06:00:00	NaN	NaN	NaN	4.673746
2021-06-10 12:00:00	NaN	NaN	NaN	4.673746
2021-06-10 18:00:00	NaN	NaN	NaN	4.673746
2021-06-11 00:00:00	NaN	NaN	NaN	3.963827

Next steps: Generate code with rmse_df | View recommended plots | New interactive sheet

Results Visualisation

```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))

# Plot each variable once
```

```
colors = {
    "Temperature (K)": "steelblue",
    "U Wind (m/s)": "orange",
    "V Wind (m/s)": "green",
    "MSLP (hPa)": "deeppink"  # Set custom color for MSLP
}

for var, rmse in rmse_results.items():
    plt.plot(ds_fc.time, rmse, label=var, marker='o', color=colors.get(var, None))

plt.title("RMSE Over Forecast Time (ForecastNet vs ERA5)")
plt.xlabel("Forecast Time")
plt.ylabel("RMSE")
plt.grid(True)
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```



Interpretation

The RMSE (Root Mean Square Error) analysis across the 10-day forecast period for four key variables—Temperature (K), U Wind (m/s), V Wind (m/s), and Mean Sea Level Pressure (MSLP in hPa) offers a clear view of how ForecastNet's predictions deviate from ERA5 reanalysis data. From the table and plot above, we observe that RMSE for temperature steadily increases over time, which is expected as forecast uncertainty grows with longer lead times. A similar upward trend is evident for both U and V wind components, suggesting that wind vector errors accumulate progressively.

Interestingly, MSLP behaves differently—it shows a relatively flat and high RMSE in the initial days (6.77 hPa), then gradually drops to around 5.1 hPa and stays flat for the remainder of the period. This flat-line behavior suggests a data artifact, possibly due to repeated values being carried forward or an issue in interpolation/resolution mismatch between datasets. It is not reflective of actual model performance change and should be re-examined with debugging if MSLP accuracy is a focal point.

In summary, while the temperature and wind forecasts show logical RMSE trends that grow over time, the MSLP's pattern may need verification. Overall, the RMSE values remain within acceptable limits for early lead times and start to rise past day 5–6, which aligns with common expectations for medium-range forecasting models.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.