# Projeto de Ciência de Dados

Professor: Luciano Barbosa

2024.1

---

Grupo:

- Camila Barbosa Vieira (cbv2)
- Luis Felipe Rodrigues de Oliveira (lfro2)

## Objetivo

O objetivo deste trabalho é analisar como as condições climáticas afetam a partida ou não dos voos nos dez maiores aeroportos do Brasil. Para isso, coletaremos dados sobre o status dos voos e as condições climáticas correspondentes durante um período de 30 dias, a partir de 7 de abril de 2024.

## Coleta de Dados

**Dados dos Voos**

Utilizaremos o site Avionio (www.avionio.com) para coletar informações sobre os voos, incluindo:

- Horário (Time)
- Data (Date)
- Código IATA do aeroporto (IATA)
- Destino (Destination)
- Número do voo (Flight)
- Companhia aérea (Airline)
- Status do voo (Status)
- Origem (Origin)

A coleta será feita por meio de um web crawler utilizando a biblioteca BeautifulSoup, que permitirá extrair essas informações das classes HTML correspondentes.

**Dados Climáticos**

Os dados climáticos serão obtidos por meio da API da Open-Meteo (https://api.open-meteo.com/v1/forecast). As variáveis climáticas a serem coletadas incluem:

- Temperatura a 2 metros (temperature_2m)
- Umidade relativa a 2 metros (relative_humidity_2m)
- Ponto de orvalho a 2 metros (dew_point_2m)
- Temperatura aparente (apparent_temperature)
- Probabilidade de precipitação (precipitation_probability)
- Precipitação (precipitation)
- Chuva (rain)
- Pancadas de chuva (showers)

- Queda de neve (snowfall)
- Pressão ao nível do mar (pressure_msl)
- Cobertura de nuvens (cloud_cover)
- Visibilidade (visibility)
- Velocidade do vento a 10 metros (wind_speed_10m)
- Direção do vento a 10 metros (wind_direction_10m)
- Rajadas de vento a 10 metros (wind_gusts_10m)

### Análise

A partir dos dados coletados, realizaremos uma análise para identificar possíveis correlações entre as condições climáticas e o status dos voos (atrasos, cancelamentos, etc.). Essa análise permitirá entender melhor como diferentes variáveis climáticas podem impactar as operações de voo nos principais aeroportos do Brasil.

## Imports e Downloads

In [ ]:
```
!pip install mlflow
!pip install optuna
!pip install lime
```

```
Collecting mlflow
  Downloading mlflow-2.14.3-py3-none-any.whl (25.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 25.8/25.8 MB 5.3 MB/s eta 0:00:00
Requirement already satisfied: Flask<4 in /usr/local/lib/python3.10/dist-packages (from
 mlflow) (2.2.5)
Collecting alembic!=1.10.0,<2 (from mlflow)
  Downloading alembic-1.13.2-py3-none-any.whl (232 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 233.0/233.0 kB 11.5 MB/s eta 0:00:00
Requirement already satisfied: cachetools<6,>=5.0.0 in /usr/local/lib/python3.10/dist-pa
ckages (from mlflow) (5.3.3)
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.10/dist-packages
 (from mlflow) (8.1.7)
Requirement already satisfied: cloudpickle<4 in /usr/local/lib/python3.10/dist-packages
 (from mlflow) (2.2.1)
Collecting docker<8,>=4.0.0 (from mlflow)
  Downloading docker-7.1.0-py3-none-any.whl (147 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 147.8/147.8 kB 7.5 MB/s eta 0:00:00
Requirement already satisfied: entrypoints<1 in /usr/local/lib/python3.10/dist-packages
 (from mlflow) (0.4)
Collecting gitpython<4,>=3.1.9 (from mlflow)
  Downloading GitPython-3.1.43-py3-none-any.whl (207 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.3/207.3 kB 13.2 MB/s eta 0:00:00
Collecting graphene<4 (from mlflow)
  Downloading graphene-3.3-py2.py3-none-any.whl (128 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 128.2/128.2 kB 5.4 MB/s eta 0:00:00
Collecting importlib-metadata!=4.7.0,<8,>=3.7.0 (from mlflow)
  Downloading importlib_metadata-7.2.1-py3-none-any.whl (25 kB)
Requirement already satisfied: markdown<4,>=3.3 in /usr/local/lib/python3.10/dist-packag
es (from mlflow) (3.6)
Requirement already satisfied: matplotlib<4 in /usr/local/lib/python3.10/dist-packages
 (from mlflow) (3.7.1)
Requirement already satisfied: numpy<2 in /usr/local/lib/python3.10/dist-packages (from
 mlflow) (1.25.2)
Collecting opentelemetry-api<3,>=1.9.0 (from mlflow)
  Downloading opentelemetry_api-1.25.0-py3-none-any.whl (59 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 59.9/59.9 kB 6.1 MB/s eta 0:00:00
Collecting opentelemetry-sdk<3,>=1.9.0 (from mlflow)
  Downloading opentelemetry_sdk-1.25.0-py3-none-any.whl (107 kB)
```

```
     □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 107.0/107.0 kB 13.4 MB/s eta 0:00:00
Requirement already satisfied: packaging<25 in /usr/local/lib/python3.10/dist-packages
 (from mlflow) (24.1)
Requirement already satisfied: pandas<3 in /usr/local/lib/python3.10/dist-packages (from
 mlflow) (2.0.3)
Requirement already satisfied: protobuf<5,>=3.12.0 in /usr/local/lib/python3.10/dist-pac
kages (from mlflow) (3.20.3)
Requirement already satisfied: pyarrow<16,>=4.0.0 in /usr/local/lib/python3.10/dist-pack
ages (from mlflow) (14.0.2)
Requirement already satisfied: pytz<2025 in /usr/local/lib/python3.10/dist-packages (fro
m mlflow) (2023.4)
Requirement already satisfied: pyyaml<7,>=5.1 in /usr/local/lib/python3.10/dist-packages
 (from mlflow) (6.0.1)
Collecting querystring-parser<2 (from mlflow)
  Downloading querystring_parser-1.2.4-py2.py3-none-any.whl (7.9 kB)
Requirement already satisfied: requests<3,>=2.17.3 in /usr/local/lib/python3.10/dist-pac
kages (from mlflow) (2.31.0)
Requirement already satisfied: scikit-learn<2 in /usr/local/lib/python3.10/dist-packages
 (from mlflow) (1.2.2)
Requirement already satisfied: scipy<2 in /usr/local/lib/python3.10/dist-packages (from
 mlflow) (1.11.4)
Requirement already satisfied: sqlalchemy<3,>=1.4.0 in /usr/local/lib/python3.10/dist-pa
ckages (from mlflow) (2.0.31)
Requirement already satisfied: sqlparse<1,>=0.4.0 in /usr/local/lib/python3.10/dist-pack
ages (from mlflow) (0.5.0)
Requirement already satisfied: Jinja2<4,>=2.11 in /usr/local/lib/python3.10/dist-package
s (from mlflow) (3.1.4)
Collecting gunicorn<23 (from mlflow)
  Downloading gunicorn-22.0.0-py3-none-any.whl (84 kB)
     □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 84.4/84.4 kB 9.1 MB/s eta 0:00:00
Collecting Mako (from alembic!=1.10.0,<2->mlflow)
  Downloading Mako-1.3.5-py3-none-any.whl (78 kB)
     □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 78.6/78.6 kB 8.5 MB/s eta 0:00:00
Requirement already satisfied: typing-extensions>=4 in /usr/local/lib/python3.10/dist-pa
ckages (from alembic!=1.10.0,<2->mlflow) (4.12.2)
Requirement already satisfied: urllib3>=1.26.0 in /usr/local/lib/python3.10/dist-package
s (from docker<8,>=4.0.0->mlflow) (2.0.7)
Requirement already satisfied: Werkzeug>=2.2.2 in /usr/local/lib/python3.10/dist-package
s (from Flask<4->mlflow) (3.0.3)
Requirement already satisfied: itsdangerous>=2.0 in /usr/local/lib/python3.10/dist-packa
ges (from Flask<4->mlflow) (2.2.0)
Collecting gitdb<5,>=4.0.1 (from gitpython<4,>=3.1.9->mlflow)
  Downloading gitdb-4.0.11-py3-none-any.whl (62 kB)
     □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 62.7/62.7 kB 7.9 MB/s eta 0:00:00
Collecting graphql-core<3.3,>=3.1 (from graphene<4->mlflow)
  Downloading graphql_core-3.2.3-py3-none-any.whl (202 kB)
     □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 202.9/202.9 kB 19.7 MB/s eta 0:00:00
Collecting graphql-relay<3.3,>=3.1 (from graphene<4->mlflow)
  Downloading graphql_relay-3.2.0-py3-none-any.whl (16 kB)
Collecting aniso8601<10,>=8 (from graphene<4->mlflow)
  Downloading aniso8601-9.0.1-py2.py3-none-any.whl (52 kB)
     □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 52.8/52.8 kB 1.7 MB/s eta 0:00:00
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.10/dist-packages (fro
m importlib-metadata!=4.7.0,<8,>=3.7.0->mlflow) (3.19.2)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-package
s (from Jinja2<4,>=2.11->mlflow) (2.1.5)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packag
es (from matplotlib<4->mlflow) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages
 (from matplotlib<4->mlflow) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packa
ges (from matplotlib<4->mlflow) (4.53.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packa
ges (from matplotlib<4->mlflow) (1.4.5)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages
 (from matplotlib<4->mlflow) (9.4.0)
```

```
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packag
es (from matplotlib<4->mlflow) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-pa
ckages (from matplotlib<4->mlflow) (2.8.2)
Collecting deprecated>=1.2.6 (from opentelemetry-api<3,>=1.9.0->mlflow)
  Downloading Deprecated-1.2.14-py2.py3-none-any.whl (9.6 kB)
Collecting importlib-metadata!=4.7.0,<8,>=3.7.0 (from mlflow)
  Downloading importlib_metadata-7.1.0-py3-none-any.whl (24 kB)
Collecting opentelemetry-semantic-conventions==0.46b0 (from opentelemetry-sdk<3,>=1.9.0-
>mlflow)
  Downloading opentelemetry_semantic_conventions-0.46b0-py3-none-any.whl (130 kB)
     □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 130.5/130.5 kB 7.8 MB/s eta 0:00:00
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages
 (from pandas<3->mlflow) (2024.1)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from quer
ystring-parser<2->mlflow) (1.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dis
t-packages (from requests<3,>=2.17.3->mlflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages
 (from requests<3,>=2.17.3->mlflow) (3.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-pack
ages (from requests<3,>=2.17.3->mlflow) (2024.6.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages
 (from scikit-learn<2->mlflow) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-pa
ckages (from scikit-learn<2->mlflow) (3.5.0)
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.10/dist-packag
es (from sqlalchemy<3,>=1.4.0->mlflow) (3.0.3)
Requirement already satisfied: wrapt<2,>=1.10 in /usr/local/lib/python3.10/dist-packages
 (from deprecated>=1.2.6->opentelemetry-api<3,>=1.9.0->mlflow) (1.14.1)
Collecting smmap<6,>=3.0.1 (from gitdb<5,>=4.0.1->gitpython<4,>=3.1.9->mlflow)
  Downloading smmap-5.0.1-py3-none-any.whl (24 kB)
Installing collected packages: aniso8601, smmap, querystring-parser, Mako, importlib-met
adata, gunicorn, graphql-core, deprecated, opentelemetry-api, graphql-relay, gitdb, dock
er, alembic, opentelemetry-semantic-conventions, graphene, gitpython, opentelemetry-sdk,
 mlflow
  Attempting uninstall: importlib-metadata
    Found existing installation: importlib_metadata 8.0.0
    Uninstalling importlib_metadata-8.0.0:
      Successfully uninstalled importlib_metadata-8.0.0
Successfully installed Mako-1.3.5 alembic-1.13.2 aniso8601-9.0.1 deprecated-1.2.14 docke
r-7.1.0 gitdb-4.0.11 gitpython-3.1.43 graphene-3.3 graphql-core-3.2.3 graphql-relay-3.2.
0 gunicorn-22.0.0 importlib-metadata-7.1.0 mlflow-2.14.3 opentelemetry-api-1.25.0 opente
lemetry-sdk-1.25.0 opentelemetry-semantic-conventions-0.46b0 querystring-parser-1.2.4 sm
map-5.0.1
Collecting optuna
  Downloading optuna-3.6.1-py3-none-any.whl (380 kB)
     □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 380.1/380.1 kB 2.2 MB/s eta 0:00:00
Requirement already satisfied: alembic>=1.5.0 in /usr/local/lib/python3.10/dist-packages
 (from optuna) (1.13.2)
Collecting colorlog (from optuna)
  Downloading colorlog-6.8.2-py3-none-any.whl (11 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from op
tuna) (1.25.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-package
s (from optuna) (24.1)
Requirement already satisfied: sqlalchemy>=1.3.0 in /usr/local/lib/python3.10/dist-packa
ges (from optuna) (2.0.31)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from opt
una) (4.66.4)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-packages (from o
ptuna) (6.0.1)
Requirement already satisfied: Mako in /usr/local/lib/python3.10/dist-packages (from ale
mbic>=1.5.0->optuna) (1.3.5)
Requirement already satisfied: typing-extensions>=4 in /usr/local/lib/python3.10/dist-pa
ckages (from alembic>=1.5.0->optuna) (4.12.2)
```

```
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.10/dist-packag
es (from sqlalchemy>=1.3.0->optuna) (3.0.3)
Requirement already satisfied: MarkupSafe>=0.9.2 in /usr/local/lib/python3.10/dist-packa
ges (from Mako->alembic>=1.5.0->optuna) (2.1.5)
Installing collected packages: colorlog, optuna
Successfully installed colorlog-6.8.2 optuna-3.6.1
Collecting shap
  Downloading shap-0.46.0-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.manylin
ux_2_17_x86_64.manylinux2014_x86_64.whl (540 kB)
     □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 540.1/540.1 kB 3.0 MB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from sh
ap) (1.25.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from sh
ap) (1.11.4)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages
 (from shap) (1.2.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from s
hap) (2.0.3)
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.10/dist-packages
 (from shap) (4.66.4)
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.10/dist-packages
 (from shap) (24.1)
Collecting slicer==0.0.8 (from shap)
  Downloading slicer-0.0.8-py3-none-any.whl (15 kB)
Requirement already satisfied: numba in /usr/local/lib/python3.10/dist-packages (from sh
ap) (0.58.1)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-packages (f
rom shap) (2.2.1)
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in /usr/local/lib/python3.10/d
ist-packages (from numba->shap) (0.41.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-
packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages
 (from pandas->shap) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages
 (from pandas->shap) (2024.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages
 (from scikit-learn->shap) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-pa
ckages (from scikit-learn->shap) (3.5.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from
 python-dateutil>=2.8.2->pandas->shap) (1.16.0)
Installing collected packages: slicer, shap
Successfully installed shap-0.46.0 slicer-0.0.8
Collecting lime
  Downloading lime-0.2.0.1.tar.gz (275 kB)
     □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ 275.7/275.7 kB 1.9 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (fr
om lime) (3.7.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from li
me) (1.25.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from li
me) (1.11.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from lim
e) (4.66.4)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.10/dist-pack
ages (from lime) (1.2.2)
Requirement already satisfied: scikit-image>=0.12 in /usr/local/lib/python3.10/dist-pack
ages (from lime) (0.19.3)
Requirement already satisfied: networkx>=2.2 in /usr/local/lib/python3.10/dist-packages
 (from scikit-image>=0.12->lime) (3.3)
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,!=8.3.0,>=6.1.0 in /usr/local/lib/p
ython3.10/dist-packages (from scikit-image>=0.12->lime) (9.4.0)
Requirement already satisfied: imageio>=2.4.1 in /usr/local/lib/python3.10/dist-packages
 (from scikit-image>=0.12->lime) (2.31.6)
```

```
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.10/dist-pac
kages (from scikit-image>=0.12->lime) (2024.6.18)
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.10/dist-packa
ges (from scikit-image>=0.12->lime) (1.6.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-package
s (from scikit-image>=0.12->lime) (24.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages
 (from scikit-learn>=0.18->lime) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-pa
ckages (from scikit-learn>=0.18->lime) (3.5.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packag
es (from matplotlib->lime) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages
 (from matplotlib->lime) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packa
ges (from matplotlib->lime) (4.53.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packa
ges (from matplotlib->lime) (1.4.5)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packag
es (from matplotlib->lime) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-pa
ckages (from matplotlib->lime) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from
 python-dateutil>=2.7->matplotlib->lime) (1.16.0)
Building wheels for collected packages: lime
  Building wheel for lime (setup.py) ... done
  Created wheel for lime: filename=lime-0.2.0.1-py3-none-any.whl size=283835 sha256=acb8
5e850b3d8db900e579096b09bb177d135fcd0460adf01ef2feb3bc3ed0ab
  Stored in directory: /root/.cache/pip/wheels/fd/a2/af/9ac0a1a85a27f314a06b39e1f492bee1
547d52549a4606ed89
Successfully built lime
Installing collected packages: lime
Successfully installed lime-0.2.0.1
```

In [3]:
```python
import re
import requests
import numpy as np
import pandas as pd
import seaborn as sns
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt
from sklearn.impute import KNNImputer
from datetime import datetime, timedelta
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import MinMaxScaler
from scipy.stats import ttest_ind, mannwhitneyu, shapiro

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.cluster import KMeans
import random
import optuna
import mlflow
import mlflow.sklearn
import lime
import lime.lime_tabular

pd.set_option('display.max_columns', None)
```

## Coleta de Dados

## Web Crawler

- Integrar dados ou extrair dados da Web

### Voos

#### Descrição

O link "https://www.avionio.com/en/airport/{cidade}/departures?ts={initial+(day*i)}&page={page}" direciona para a página correta do Avionio, permitindo a coleta dos dados dos voos. Para isso, é necessário substituir:

- **{cidade}**: pela sigla IATA do aeroporto de partida.
- **{initial}**: pelo timestamp correspondente ao dia inicial do período desejado.
- **{day}**: pelo valor representativo de um dia em milissegundos (86400000 ms).
- **{i}**: pelo número de dias a partir da data inicial.
- **{page}**: pelo número da página, começando do zero.

### Detalhamento

1. **Sigla IATA do Aeroporto**:
    - Substitua **{cidade}** pela sigla IATA do aeroporto de partida. Por exemplo, "GRU" para o Aeroporto Internacional de São Paulo/Guarulhos.
2. **Timestamp do Dia Inicial**:
    - **{initial}** representa o timestamp do dia inicial do período. Por exemplo, para 7 de abril de 2024, você precisa converter essa data para timestamp em milissegundos.
3. **Incremento Diário**:
    - **{day}** é o incremento diário de 86400000 milissegundos, equivalente a um dia. Para avançar para o próximo dia, você adiciona esse valor ao timestamp inicial.
4. **Número do Dia**:
    - **{i}** é o número do dia a partir da data inicial. Para o dia inicial, **{i}** é 0; para o segundo dia, **{i}** é 1, e assim por diante.
5. **Número da Página**:
    - **{page}** representa o número da página de resultados, começando do zero. Para obter todos os voos de um dia, é necessário iterar pelas páginas até que todos os dados sejam coletados.

### Exemplo

Para coletar dados de partidas do Aeroporto Internacional de São Paulo/Guarulhos (GRU) no dia 7 de abril de 2024:

1. **Sigla do aeroporto**: "GRU"
2. **Timestamp do dia inicial**: suponha que seja "1712484000000".
3. **Valor diário em milissegundos**: 86400000
4. **Número do dia (i)**: 0 (para o dia inicial)
5. **Página**: 0 (primeira página de resultados)

O link ficaria:

```
https://www.avionio.com/en/airport/GRU/departures?
ts=1712484000000&page=0
```

Para coletar dados do dia seguinte (8 de abril de 2024), o timestamp seria incrementado por 86400000:

```
https://www.avionio.com/en/airport/GRU/departures?
ts=1712570400000&page=0
```

Continuando dessa maneira, é possível coletar os dados para cada dia do período de 30 dias, alterando o valor de **{i}** e o timestamp correspondente. Para cada dia, itere pelas páginas até que todos os voos sejam coletados.

```
In [ ]: initial = 1712484000000 #timestamp para o dia 07 Apr
        day = 86400000          #timestamp para duração de um dia
        total_days = 30         #total de dias coletados
```

```
In [ ]: cidades = ['GRU', 'CGH', 'BSB', 'GIG', 'CNF', 'VCP', 'SDU', 'REC', 'POA', 'SSA']
        categorias = ["Time", "Date", "IATA code", "Destination", "Flight", "Airline", "Status"]
        classes = ['tt-t', 'tt-d', 'tt-i', 'tt-ap', 'tt-f', 'tt-al', 'tt-s']
        regex = r"([a-zA-Z0-9: ])"

        df = pd.DataFrame()

        for cidade in cidades:
          print(cidade)
          for i in range(total_days):
            data = {}
            date = datetime.strptime('07 Apr 2024', '%d %b %Y') + timedelta(days=i)
            previous_date = datetime.strptime('07 Apr 2024', '%d %b %Y') + timedelta(days=i)
            next_date = datetime.strptime('07 Apr 2024', '%d %b %Y') + timedelta(days=i)
            page = 0

            while(previous_date == date):
              url = f"https://www.avionio.com/en/airport/{cidade}/departures?ts={initial + (day*
              response = requests.get(url)
              if response.status_code == 200:
                html_content = response.content
                soup = BeautifulSoup(html_content, 'lxml')

                for classe, categoria in zip(classes, categorias):
                  data_elements = soup.find_all('td', class_= classe)
                  data[categoria] = []
                  for element in data_elements:
                    elements = []
                    description = element.text
                    elements.append(re.findall(regex, description))
                    data[categoria].append(("".join(elements[0]).strip()))

                day_df = (pd.DataFrame(data))

                day_df["Origin"] = cidade
                df = pd.concat([df, day_df], ignore_index=True)

                date_string = data['Date'][-1] + ' 2024'
                previous_date = datetime.strptime(date_string, '%d %b %Y')
                page -= 1
              else:
                print(f"Erro ao acessar o site: {response.status_code}")
```

```
    page = 1
    while(next_date == date):
      url = f"https://www.avionio.com/en/airport/{cidade}/departures?ts={initial + (day*
      response = requests.get(url)
      if response.status_code == 200:
        html_content = response.content
        soup = BeautifulSoup(html_content, 'lxml')

        for classe, categoria in zip(classes, categorias):
          data_elements = soup.find_all('td', class_= classe)
          data[categoria] = []
          for element in data_elements:
            elements = []
            description = element.text
            elements.append(re.findall(regex, description))
            data[categoria].append(("".join(elements[0]).strip()))

        day_df = (pd.DataFrame(data))
        day_df["Origin"] = cidade
        df = pd.concat([df, day_df], ignore_index=True)

        date_string = data['Date'][-1] + ' 2024'
        next_date = datetime.strptime(date_string, '%d %b %Y')
        page += 1
      else:
        print(f"Erro ao acessar o site: {response.status_code}")
df
```

```
GRU
CGH
BSB
GIG
CNF
VCP
SDU
REC
POA
SSA
```

Out[ ]:

| | Time | Date | IATA code | Destination | Flight | Airline | Status | Origin |
|---|---|---|---|---|---|---|---|---|
| **0** | 13:00 | 07 Apr | CWB | Curitiba | LA3286 | LATAM Airlines 2 | Departed 13:37 | GRU |
| **1** | 13:00 | 07 Apr | CWB | Curitiba | DL7371 | Delta Air Lines | Departed 13:37 | GRU |
| **2** | 13:00 | 07 Apr | CWB | Curitiba | QR5117 | Qatar Airways | Departed 13:37 | GRU |
| **3** | 13:05 | 07 Apr | MAO | Manaus | G31606 | Gol 6 | Departed 13:19 | GRU |
| **4** | 13:05 | 07 Apr | MAO | Manaus | AA7689 | American Airlines | Departed 13:19 | GRU |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **130533** | 02:45 | 07 May | VCP | Campinas | TP5324 | TAP Air Portugal | Departed 02:46 | SSA |
| **130534** | 03:30 | 07 May | GIG | Rio De Janeiro | LA3673 | LATAM Airlines 2 | Departed 03:25 | SSA |
| **130535** | 03:30 | 07 May | GIG | Rio De Janeiro | DL6322 | Delta Air Lines | Departed 03:25 | SSA |
| **130536** | 03:30 | 07 May | GIG | Rio De Janeiro | LH4679 | Lufthansa | Departed 03:25 | SSA |
| **130537** | 05:00 | 07 May | CGH | Sao Paulo | LA3623 | LATAM Airlines | Departed 05:04 | SSA |

130538 rows × 8 columns

In [ ]:
```
df.to_csv('/content/avionio.csv', index=False)
```

In [ ]:
```
len(df['IATA code'].unique())
```

Out[ ]:
```
162
```

In [ ]:
```
df['IATA code'].unique()
```

Out[ ]:
```
array(['CWB', 'MAO', 'MDZ', 'SCL', 'CXJ', 'CNF', 'JPA', 'POA', 'AEP',
       'MCZ', 'AJU', 'MAD', 'REC', 'SSA', 'VVI', 'GIG', 'SLZ', 'NAT',
       'BPS', 'VIX', 'FOR', 'CGB', 'IGU', 'GYN', 'BSB', 'UIO', 'JJD',
       'RAO', 'NVT', 'FLN', 'BEL', 'PDP', 'LDB', 'ASU', 'ATL', 'SDU',
       'MIA', 'PTY', 'MCO', 'IOS', 'MVD', 'CGR', 'CAC', 'MEX', 'THE',
       'IZA', 'RVD', 'OPS', 'MGF', 'UDI', 'MOC', 'JJG', 'PNZ', 'LIM',
       'XAP', 'PMW', 'SJP', 'JOI', 'BOG', 'ADD', 'EZE', 'IST', 'SDQ',
       'FRA', 'VDC', 'LHR', 'LIS', 'DXB', 'DOH', 'JDO', 'JFK', 'LAX',
       'IMP', 'FCO', 'PFB', 'CDG', 'PPB', 'PET', 'MXP', 'BCN', 'LAD',
       'ZRH', 'YYZ', 'EWR', 'IAD', 'AMS', 'IAH', 'DFW', 'ORD', 'BOS',
       'AAX', 'JNB', 'VCP', 'YUL', 'UNA', 'DSS', 'GEL', 'OPO', 'PVH',
       'CPT', 'PUJ', 'CGH', 'BEY', 'COR', 'RRJ', 'UBA', 'CLV', 'IPN',
       'ARU', 'BYO', 'GRU', 'BVB', 'AUX', 'RBR', 'MCP', 'MAB', 'STM',
       'BRA', 'FLL', 'ROS', 'CPV', 'SJK', 'LUX', 'TFL', 'GVR', 'CKS',
       'LEC', 'LHN', 'VAG', 'JMA', 'CFB', 'GNM', 'POJ', 'CUR', 'ATM',
       'CCP', 'PGZ', 'PMG', 'MII', 'TJL', 'JTC', 'CMG', 'ORY', 'CAW',
       'ROO', 'GPB', 'MDE', 'MEM', 'SJU', 'MVF', 'FEN', 'FEC', 'PAV',
       'CAU', 'SET', 'LPA', 'RIA', 'URG', 'BGX', 'CSU', 'SRA', 'ALQ'],
      dtype=object)
```

## Clima

A partir dos aeroportos listados no DataFrame de voos, foi criada uma lista de dicionários contendo a sigla do aeroporto, bem como suas coordenadas de latitude e longitude. Posteriormente, utilizaremos essas informações para recuperar os dados climáticos de todas essas localidades durante o período especificado. O objetivo final é unir esses dados climáticos com a tabela de voos, criando assim um conjunto completo de informações que relacionam voos e condições climáticas.

In [ ]:
```
latitudes_longitudes = [
  {"sigla": "CWB", "latitude": -25.5285, "longitude": -49.1758},
  {"sigla": "MAO", "latitude": -3.0386, "longitude": -60.0497},
  {"sigla": "MDZ", "latitude": -32.832, "longitude": -68.8272},
  {"sigla": "SCL", "latitude": -33.393, "longitude": -70.7858},
  {"sigla": "CXJ", "latitude": -29.1971, "longitude": -51.1875},
  {"sigla": "CNF", "latitude": -19.6244, "longitude": -43.9719},
  {"sigla": "JPA", "latitude": -7.1458, "longitude": -34.9489},
  {"sigla": "POA", "latitude": -29.9939, "longitude": -51.1711},
  {"sigla": "AEP", "latitude": -34.558, "longitude": -58.4155},
  {"sigla": "MCZ", "latitude": -9.5108, "longitude": -35.7917},
  {"sigla": "AJU", "latitude": -10.9853, "longitude": -37.0703},
  {"sigla": "MAD", "latitude": 40.4983, "longitude": -3.5676},
  {"sigla": "REC", "latitude": -8.1264, "longitude": -34.9236},
  {"sigla": "SSA", "latitude": -12.9086, "longitude": -38.3225},
  {"sigla": "VVI", "latitude": -17.6447, "longitude": -63.1354},
  {"sigla": "GIG", "latitude": -22.8089, "longitude": -43.2436},
  {"sigla": "SLZ", "latitude": -2.5863, "longitude": -44.2369},
  {"sigla": "NAT", "latitude": -5.7681, "longitude": -35.3761},
  {"sigla": "VIX", "latitude": -20.2581, "longitude": -40.2865},
  {"sigla": "BPS", "latitude": -16.4397, "longitude": -39.0808},
  {"sigla": "CGB", "latitude": -15.6529, "longitude": -56.1172},
  {"sigla": "IGU", "latitude": -25.5951, "longitude": -54.4872},
  {"sigla": "FOR", "latitude": -3.7763, "longitude": -38.5326},
  {"sigla": "GYN", "latitude": -16.6294, "longitude": -49.2263},
  {"sigla": "BSB", "latitude": -15.8711, "longitude": -47.9186},
  {"sigla": "UIO", "latitude": -0.1292, "longitude": -78.3579},
  {"sigla": "JJD", "latitude": -2.898, "longitude": -40.3516},
  {"sigla": "NVT", "latitude": -26.8835, "longitude": -48.656},
  {"sigla": "RAO", "latitude": -21.1342, "longitude": -47.7743},
  {"sigla": "FLN", "latitude": -27.6705, "longitude": -48.5528},
  {"sigla": "BEL", "latitude": -1.3813, "longitude": -48.4781},
  {"sigla": "PDP", "latitude": -34.8551, "longitude": -55.0943},
  {"sigla": "LDB", "latitude": -23.3331, "longitude": -51.1342},
  {"sigla": "ASU", "latitude": -25.2398, "longitude": -57.5191},
  {"sigla": "ATL", "latitude": 33.6367, "longitude": -84.4281},
  {"sigla": "SDU", "latitude": -22.9105, "longitude": -43.1631},
  {"sigla": "MIA", "latitude": 25.7959, "longitude": -80.2871},
  {"sigla": "PTY", "latitude": 9.0714, "longitude": -79.3835},
  {"sigla": "MCO", "latitude": 28.4294, "longitude": -81.3089},
  {"sigla": "IOS", "latitude": -14.815, "longitude": -39.0331},
  {"sigla": "MVD", "latitude": -34.8384, "longitude": -56.0308},
  {"sigla": "CAC", "latitude": -25.0014, "longitude": -53.5017},
  {"sigla": "CGR", "latitude": -20.4687, "longitude": -54.6728},
  {"sigla": "MEX", "latitude": 19.4361, "longitude": -99.0719},
  {"sigla": "THE", "latitude": -5.0594, "longitude": -42.8235},
  {"sigla": "IZA", "latitude": -21.5131, "longitude": -43.1733},
  {"sigla": "RVD", "latitude": -17.7241, "longitude": -50.9141},
  {"sigla": "OPS", "latitude": -11.8858, "longitude": -55.5861},
  {"sigla": "MGF", "latitude": -23.478, "longitude": -52.0122},
  {"sigla": "UDI", "latitude": -18.8836, "longitude": -48.2251},
  {"sigla": "MOC", "latitude": -16.7069, "longitude": -43.8189},
  {"sigla": "JJG", "latitude": -29.3033, "longitude": -49.0533},
  {"sigla": "LIM", "latitude": -12.0219, "longitude": -77.1143},
  {"sigla": "PNZ", "latitude": -9.3624, "longitude": -40.5691},
  {"sigla": "XAP", "latitude": -27.1342, "longitude": -52.6566},
  {"sigla": "PMW", "latitude": -10.2915, "longitude": -48.3566},
  {"sigla": "SJP", "latitude": -20.8166, "longitude": -49.4065},
  {"sigla": "JOI", "latitude": -26.2231, "longitude": -48.7973},
  {"sigla": "BOG", "latitude": 4.7016, "longitude": -74.1469},
  {"sigla": "ADD", "latitude": 8.9779, "longitude": 38.7993},
  {"sigla": "EZE", "latitude": -34.8222, "longitude": -58.5358},
  {"sigla": "IST", "latitude": 41.2753, "longitude": 28.7519},
  {"sigla": "SDQ", "latitude": 18.4294, "longitude": -69.6689},
  {"sigla": "FRA", "latitude": 50.0379, "longitude": 8.5622},
  {"sigla": "VDC", "latitude": -14.8613, "longitude": -40.8631},
  {"sigla": "LHR", "latitude": 51.4716, "longitude": -0.4643},
  {"sigla": "LIS", "latitude": 38.7742, "longitude": -9.1342},
  {"sigla": "DXB", "latitude": 25.2532, "longitude": 55.3657},
  {"sigla": "DOH", "latitude": 25.2736, "longitude": 51.6086},
  {"sigla": "JDO", "latitude": -7.2189, "longitude": -39.2701},
  {"sigla": "JFK", "latitude": 40.6413, "longitude": -73.7781},
  {"sigla": "LAX", "latitude": 33.9425, "longitude": -118.407},
  {"sigla": "IMP", "latitude": -5.5313, "longitude": -47.4594},
  {"sigla": "FCO", "latitude": 41.8003, "longitude": 12.2389},
  {"sigla": "PFB", "latitude": -28.2433, "longitude": -52.3264},
  {"sigla": "CDG", "latitude": 49.0097, "longitude": 2.5479},
  {"sigla": "PPB", "latitude": -22.1751, "longitude": -51.4246},
  {"sigla": "PET", "latitude": -31.7183, "longitude": -52.3274},
  {"sigla": "MXP", "latitude": 45.6301, "longitude": 8.7231},
  {"sigla": "BCN", "latitude": 41.2974, "longitude": 2.0833},
  {"sigla": "LAD", "latitude": -8.8583, "longitude": 13.2312},
  {"sigla": "ZRH", "latitude": 47.4582, "longitude": 8.5481},
  {"sigla": "YYZ", "latitude": 43.6777, "longitude": -79.6248},
  {"sigla": "EWR", "latitude": 40.6895, "longitude": -74.1745},
  {"sigla": "IAD", "latitude": 38.9531, "longitude": -77.4565},
  {"sigla": "AMS", "latitude": 52.3105, "longitude": 4.7683},
  {"sigla": "IAH", "latitude": 29.9902, "longitude": -95.3368},
  {"sigla": "DFW", "latitude": 32.8998, "longitude": -97.0403},
  {"sigla": "ORD", "latitude": 41.9742, "longitude": -87.9073},
  {"sigla": "BOS", "latitude": 42.3656, "longitude": -71.0096},
  {"sigla": "AAX", "latitude": -19.5639, "longitude": -46.9643},
  {"sigla": "JNB", "latitude": -26.1337, "longitude": 28.2426},
  {"sigla": "VCP", "latitude": -23.0067, "longitude": -47.1344},
  {"sigla": "YUL", "latitude": 45.4577, "longitude": -73.7499},
  {"sigla": "UNA", "latitude": -15.3552, "longitude": -38.9997},
  {"sigla": "DSS", "latitude": 14.6711, "longitude": -17.0711},
  {"sigla": "GEL", "latitude": -28.2825, "longitude": -54.1691},
  {"sigla": "OPO", "latitude": 41.2481, "longitude": -8.6814},
  {"sigla": "PVH", "latitude": -8.709, "longitude": -63.9023},
  {"sigla": "CPT", "latitude": -33.9648, "longitude": 18.6017},
  {"sigla": "PUJ", "latitude": 18.5674, "longitude": -68.3634},
  {"sigla": "CGH", "latitude": -23.6261, "longitude": -46.6564},
  {"sigla": "BEY", "latitude": 33.8209, "longitude": 35.4883},
  {"sigla": "COR", "latitude": -31.3236, "longitude": -64.2144},
  {"sigla": "RRJ", "latitude": -27.4874, "longitude": -49.1746},
  {"sigla": "UBA", "latitude": -19.7647, "longitude": -47.9644},
  {"sigla": "CLV", "latitude": -17.7255, "longitude": -48.6073},
  {"sigla": "IPN", "latitude": -19.4705, "longitude": -42.4876},
  {"sigla": "ARU", "latitude": -21.1413, "longitude": -50.4247},
  {"sigla": "BYO", "latitude": -21.2473, "longitude": -56.4525},
  {"sigla": "GRU", "latitude": -23.4356, "longitude": -46.4731},
  {"sigla": "BVB", "latitude": 2.8489, "longitude": -60.6901},
  {"sigla": "AUX", "latitude": -7.2279, "longitude": -48.2405},
  {"sigla": "RBR", "latitude": -9.8689, "longitude": -67.898},
  {"sigla": "MCP", "latitude": 0.0506, "longitude": -51.0722},
  {"sigla": "MAB", "latitude": -5.3686, "longitude": -49.1381},
  {"sigla": "STM", "latitude": -2.4247, "longitude": -54.7856},
  {"sigla": "BRA", "latitude": -12.0827, "longitude": -45.0084},
  {"sigla": "FLL", "latitude": 26.0726, "longitude": -80.1527},
  {"sigla": "ROS", "latitude": -32.9037, "longitude": -60.785},
  {"sigla": "CPV", "latitude": -7.2699, "longitude": -35.8964},
  {"sigla": "SJK", "latitude": -23.2292, "longitude": -45.8615},
  {"sigla": "LUX", "latitude": 49.6233, "longitude": 6.2042},
  {"sigla": "TFL", "latitude": -17.891, "longitude": -41.5136},
  {"sigla": "GVR", "latitude": -18.8953, "longitude": -41.9822},
  {"sigla": "CKS", "latitude": -6.1153, "longitude": -50.0017},
  {"sigla": "LEC", "latitude": -12.4823, "longitude": -41.2773},
  {"sigla": "LHN", "latitude": -13.2681, "longitude": -44.9488},
  {"sigla": "VAG", "latitude": -21.5563, "longitude": -45.4756},
  {"sigla": "JMA", "latitude": -21.6699, "longitude": -45.4444},
  {"sigla": "CFB", "latitude": -21.5903, "longitude": -45.4731},
  {"sigla": "GNM", "latitude": -14.2175, "longitude": -42.7808},
  {"sigla": "POJ", "latitude": -18.6696, "longitude": -46.4905},
  {"sigla": "CUR", "latitude": 12.1889, "longitude": -68.9598},
  {"sigla": "ATM", "latitude": -3.2539, "longitude": -52.2548},
  {"sigla": "CCP", "latitude": -36.7727, "longitude": -73.0631},
  {"sigla": "PGZ", "latitude": -27.1591, "longitude": -49.2235},
  {"sigla": "PMG", "latitude": -22.5487, "longitude": -55.7036},
  {"sigla": "MII", "latitude": -22.1969, "longitude": -49.9264},
  {"sigla": "TJL", "latitude": -22.7371, "longitude": -46.9864},
  {"sigla": "JTC", "latitude": -22.1572, "longitude": -49.0681},
  {"sigla": "CMG", "latitude": -19.0119, "longitude": -57.6724},
  {"sigla": "ORY", "latitude": 48.7253, "longitude": 2.359},
  {"sigla": "CAW", "latitude": -21.6983, "longitude": -41.3017},
  {"sigla": "ROO", "latitude": -16.5863, "longitude": -54.7243},
  {"sigla": "GPB", "latitude": -25.3875, "longitude": -51.5202},
  {"sigla": "MDE", "latitude": 6.1645, "longitude": -75.4231},
  {"sigla": "MEM", "latitude": 35.0425, "longitude": -89.9767},
  {"sigla": "SJU", "latitude": 18.4394, "longitude": -66.0018},
  {"sigla": "MVF", "latitude": -5.2025, "longitude": -37.3641},
  {"sigla": "FEN", "latitude": -3.8549, "longitude": -32.4233},
  {"sigla": "FEC", "latitude": -12.2035, "longitude": -38.9067},
  {"sigla": "PAV", "latitude": -9.401, "longitude": -40.4897},
  {"sigla": "CAU", "latitude": -8.2824, "longitude": -36.0132},
  {"sigla": "SET", "latitude": -23.948, "longitude": -46.3335},
  {"sigla": "LPA", "latitude": 27.9319, "longitude": -15.3866},
  {"sigla": "RIA", "latitude": -29.378, "longitude": -53.7188},
  {"sigla": "URG", "latitude": -29.7822, "longitude": -57.0382},
  {"sigla": "BGX", "latitude": -31.3905, "longitude": -54.1122},
  {"sigla": "CSU", "latitude": -29.7114, "longitude": -53.6882},
  {"sigla": "SRA", "latitude": -27.9067, "longitude": -54.5203},
  {"sigla": "ALQ", "latitude": -29.7865, "longitude": -57.0368}
]
```

In [ ]:
```
def apiOpenMeteo(latitude, longitude, sigla):
    hourly_data = {}
    url = f"https://archive-api.open-meteo.com/v1/era5?latitude={latitude}&longitude={lo
    hourly_categories = ['temperature_2m','relative_humidity_2m','dew_point_2m','apparen

    response = requests.get(url)
    hourly = response.json()['hourly']
    timestamps = hourly['time']
    hourly_data['timestamp'] = timestamps
    hourly_data['sigla'] = [sigla] * len(timestamps)

    for hourly_category in hourly_categories:
        hourly_data[hourly_category] = hourly[hourly_category]


    hourly_dataframe = pd.DataFrame(data=hourly_data)

    return hourly_dataframe
```

In [ ]:
```
dfs_clima = []

for loc in latitudes_longitudes:
    df_clima = apiOpenMeteo(loc['latitude'], loc['longitude'], loc['sigla'])
    dfs_clima.append(df_clima)

df_clima = pd.concat(dfs_clima, ignore_index=True)
df_clima
```

Out[ ]:

| | timestamp | sigla | temperature_2m | relative_humidity_2m | dew_point_2m | apparent_temperature | precipita |
|---|---|---|---|---|---|---|---|
| **0** | 2024-04-07T00:00 | CWB | 16.4 | 88 | 14.4 | 15.2 | |
| **1** | 2024-04-07T01:00 | CWB | 16.0 | 87 | 13.8 | 15.0 | |
| **2** | 2024-04-07T02:00 | CWB | 15.5 | 90 | 13.9 | 15.0 | |
| **3** | 2024-04-07T03:00 | CWB | 14.7 | 94 | 13.8 | 14.6 | |
| **4** | 2024-04-07T04:00 | CWB | 14.3 | 94 | 13.3 | 14.3 | |
| **...** | ... | ... | ... | ... | ... | ... | |
| **120523** | 2024-05-07T19:00 | ALQ | 29.3 | 66 | 22.4 | 31.5 | |
| **120524** | 2024-05-07T20:00 | ALQ | 28.6 | 68 | 22.2 | 30.7 | |
| **120525** | 2024-05-07T21:00 | ALQ | 27.9 | 69 | 21.7 | 29.7 | |
| **120526** | 2024-05-07T22:00 | ALQ | 27.4 | 69 | 21.3 | 28.8 | |
| **120527** | 2024-05-07T23:00 | ALQ | 26.8 | 71 | 21.2 | 28.5 | |

120528 rows × 17 columns

In [ ]:
```
df_clima.to_csv('/content/clima.csv', index=False)
```

Junção

```
In [ ]: df_voo = pd.read_csv('/content/avionio.csv')
        df_clima = pd.read_csv('/content/clima.csv')
```

```
In [ ]: df_voo['datetime'] = pd.to_datetime(df_voo['Date'] + ' 2024 ' + df_voo['Time'].apply(lam
        df_clima['datetime'] = pd.to_datetime(df_clima['timestamp'])
```

```
In [ ]: dfComplete = pd.merge(df_voo, df_clima, left_on=['datetime', 'Origin'], right_on=['datet
        dfComplete = dfComplete.drop(columns=['sigla', 'timestamp'])
        dfComplete = pd.merge(dfComplete, df_clima, left_on=['datetime', 'IATA code'], right_on=
        dfComplete = dfComplete.drop(columns=['sigla', 'Time', 'Date', 'timestamp'])
        dfComplete
```

Out[ ]:

| | IATA code | Destination | Flight | Airline | Status | Origin | datetime | temperature_2m | relative_humidity_2r |
|---|---|---|---|---|---|---|---|---|---|
| **0** | CWB | Curitiba | LA3286 | LATAM Airlines 2 | Departed 13:37 | GRU | 2024-04-07 13:00:00 | 22.6 | 7 |
| **1** | CWB | Curitiba | DL7371 | Delta Air Lines | Departed 13:37 | GRU | 2024-04-07 13:00:00 | 22.6 | 7 |
| **2** | CWB | Curitiba | QR5117 | Qatar Airways | Departed 13:37 | GRU | 2024-04-07 13:00:00 | 22.6 | 7 |
| **3** | CWB | Curitiba | G31106 | Gol | Departed 13:23 | CGH | 2024-04-07 13:00:00 | 22.5 | 7 |
| **4** | CWB | Curitiba | LA3248 | LATAM Airlines | Departed 14:03 | CGH | 2024-04-07 13:00:00 | 22.5 | 7 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | . |
| **129900** | VCP | Campinas | AD4027 | Azul 1 | Departed 02:46 | SSA | 2024-05-07 02:00:00 | 26.7 | 7 |
| **129901** | VCP | Campinas | TP5324 | TAP Air Portugal | Departed 02:46 | SSA | 2024-05-07 02:00:00 | 26.7 | 7 |
| **129902** | GIG | Rio De Janeiro | LA3673 | LATAM Airlines 2 | Departed 03:25 | SSA | 2024-05-07 03:00:00 | 26.8 | 7 |
| **129903** | GIG | Rio De Janeiro | DL6322 | Delta Air Lines | Departed 03:25 | SSA | 2024-05-07 03:00:00 | 26.8 | 7 |
| **129904** | GIG | Rio De Janeiro | LH4679 | Lufthansa | Departed 03:25 | SSA | 2024-05-07 03:00:00 | 26.8 | 7 |

129905 rows × 37 columns

```
In [ ]: dfComplete.to_csv('/content/dfComplete.csv', index=False)
```

## Estatísticas Descritivas

- Visualizações

```
In [ ]: dfComplete = pd.read_csv('/content/dfComplete.csv')
        dfComplete
```

Out[ ]:

| | IATA code | Destination | Flight | Airline | Status | Origin | datetime | temperature_2m | relative_humidity_2r |
|---|---|---|---|---|---|---|---|---|---|
| **0** | CWB | Curitiba | LA3286 | LATAM Airlines 2 | Departed 13:37 | GRU | 2024-04-07 13:00:00 | 22.6 | 7 |
| **1** | CWB | Curitiba | DL7371 | Delta Air Lines | Departed 13:37 | GRU | 2024-04-07 13:00:00 | 22.6 | 7 |
| **2** | CWB | Curitiba | QR5117 | Qatar Airways | Departed 13:37 | GRU | 2024-04-07 13:00:00 | 22.6 | 7 |
| **3** | CWB | Curitiba | G31106 | Gol | Departed 13:23 | CGH | 2024-04-07 13:00:00 | 22.5 | 7 |
| **4** | CWB | Curitiba | LA3248 | LATAM Airlines | Departed 14:03 | CGH | 2024-04-07 13:00:00 | 22.5 | 7 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | . |
| **129900** | VCP | Campinas | AD4027 | Azul 1 | Departed 02:46 | SSA | 2024-05-07 02:00:00 | 26.7 | 7 |
| **129901** | VCP | Campinas | TP5324 | TAP Air Portugal | Departed 02:46 | SSA | 2024-05-07 02:00:00 | 26.7 | 7 |
| **129902** | GIG | Rio De Janeiro | LA3673 | LATAM Airlines 2 | Departed 03:25 | SSA | 2024-05-07 03:00:00 | 26.8 | 7 |
| **129903** | GIG | Rio De Janeiro | DL6322 | Delta Air Lines | Departed 03:25 | SSA | 2024-05-07 03:00:00 | 26.8 | 7 |
| **129904** | GIG | Rio De Janeiro | LH4679 | Lufthansa | Departed 03:25 | SSA | 2024-05-07 03:00:00 | 26.8 | 7 |

129905 rows × 37 columns

```
In [ ]: dfComplete.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129905 entries, 0 to 129904
Data columns (total 37 columns):
 #   Column                     Non-Null Count   Dtype  
---  ------                     --------------   -----  
 0   IATA code                  129905 non-null  object 
 1   Destination                129905 non-null  object 
 2   Flight                     129905 non-null  object 
 3   Airline                    129755 non-null  object 
 4   Status                     129905 non-null  object 
 5   Origin                     129905 non-null  object 
 6   datetime                   129905 non-null  object 
 7   temperature_2m             129905 non-null  float64
 8   relative_humidity_2m       129905 non-null  int64  
 9   dew_point_2m               129905 non-null  float64
 10  apparent_temperature       129905 non-null  float64
 11  precipitation_probability  0 non-null       float64
 12  precipitation              129905 non-null  float64
 13  rain                           129905 non-null  float64
 14  showers                        0 non-null       float64
 15  snowfall                       129905 non-null  float64
 16  pressure_msl                   129905 non-null  float64
 17  cloud_cover                    129905 non-null  int64
 18  visibility                     0 non-null       float64
 19  wind_speed_10m                 129905 non-null  float64
 20  wind_direction_10m             129905 non-null  int64
 21  wind_gusts_10m                 129905 non-null  float64
 22  temperature_2m_dst             129905 non-null  float64
 23  relative_humidity_2m_dst       129905 non-null  int64
 24  dew_point_2m_dst               129905 non-null  float64
 25  apparent_temperature_dst       129905 non-null  float64
 26  precipitation_probability_dst  0 non-null       float64
 27  precipitation_dst              129905 non-null  float64
 28  rain_dst                       129905 non-null  float64
 29  showers_dst                    0 non-null       float64
 30  snowfall_dst                   129905 non-null  float64
 31  pressure_msl_dst               129905 non-null  float64
 32  cloud_cover_dst                129905 non-null  int64
 33  visibility_dst                 0 non-null       float64
 34  wind_speed_10m_dst             129905 non-null  float64
 35  wind_direction_10m_dst         129905 non-null  int64
 36  wind_gusts_10m_dst             129905 non-null  float64
dtypes: float64(24), int64(6), object(7)
memory usage: 36.7+ MB
```

Percebe-se que, exceto pelas colunas vazias, não há muitos dados ausentes e há poucas variáveis categóricas. Isso é relevante porque a ausência de dados incompletos facilita a análise estatística e a modelagem preditiva, reduzindo a necessidade de técnicas de imputação ou de exclusão de amostras. Além disso, a predominância de variáveis numéricas simplifica a aplicação de algoritmos de aprendizado de máquina, que geralmente requerem menos pré-processamento comparado às variáveis categóricas.

In [ ]:
```
dfComplete.describe()
```

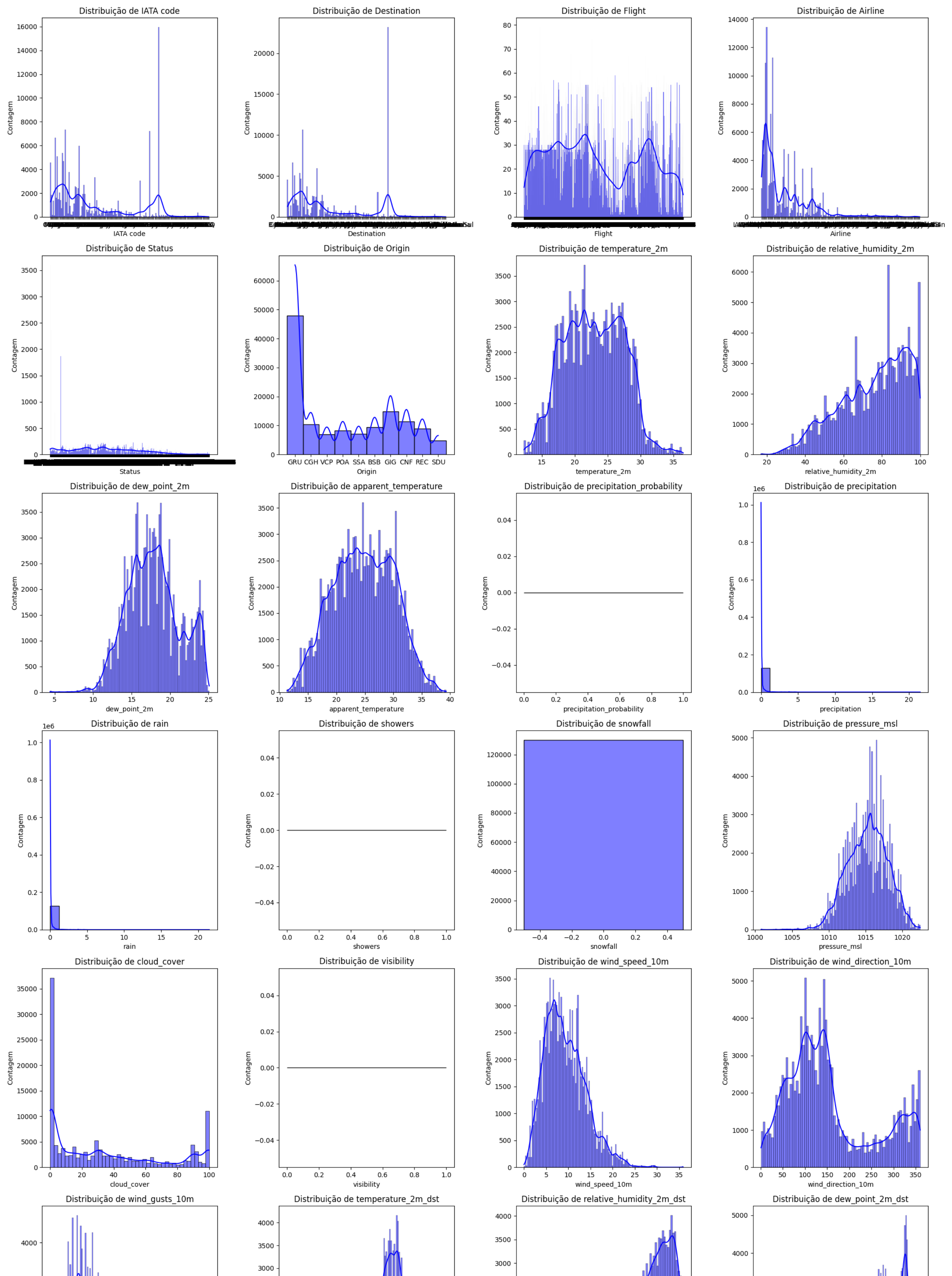
Out[ ]:

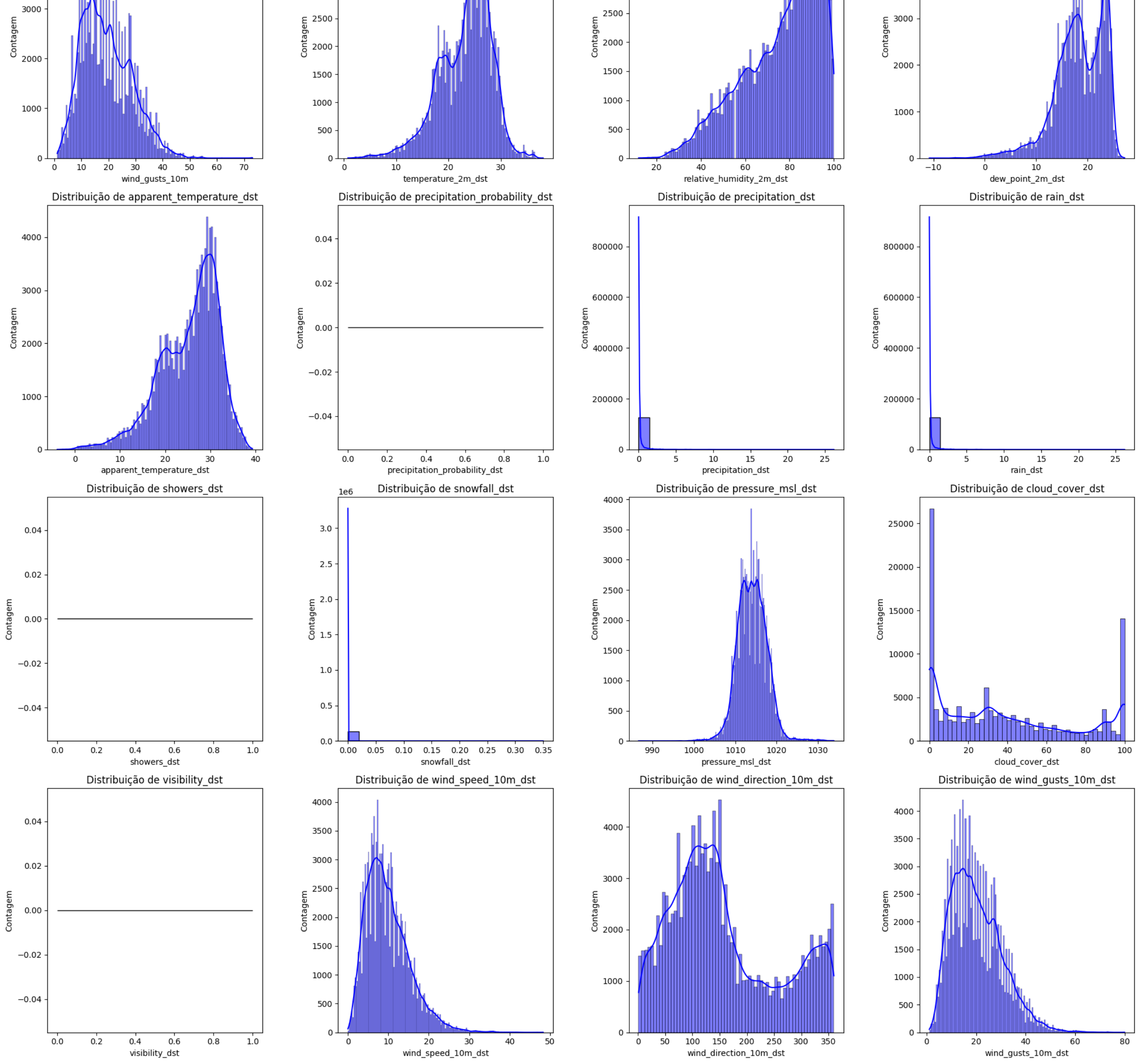
| | temperature_2m | relative_humidity_2m | dew_point_2m | apparent_temperature | precipitation_probability | |
|---|---|---|---|---|---|---|
| **count** | 129905.000000 | 129905.000000 | 129905.000000 | 129905.000000 | 0.0 | 12 |
| **mean** | 22.938524 | 75.454925 | 17.832934 | 24.735901 | NaN | |
| **std** | 4.416751 | 17.448167 | 3.353846 | 5.295025 | NaN | |
| **min** | 12.300000 | 17.000000 | 4.400000 | 11.300000 | NaN | |
| **25%** | 19.500000 | 63.000000 | 15.400000 | 20.700000 | NaN | |
| **50%** | 22.900000 | 79.000000 | 17.700000 | 24.700000 | NaN | |
| **75%** | 26.500000 | 90.000000 | 20.000000 | 28.900000 | NaN | |
| **max** | 36.500000 | 100.000000 | 25.100000 | 39.300000 | NaN | |

Na análise inicial, é possível identificar colunas vazias que podem ser eliminadas posteriormente. Observamos que, na maioria dos casos, os valores da mediana são próximos aos valores da média, indicando uma baixa presença de outliers.

## Distribuição das Variáveis

In [ ]:
```
fig, axs = plt.subplots(9, 4, figsize=(20, 5 * 9))
for i, coluna in enumerate(dfComplete.loc[:, dfComplete.columns != 'datetime'].columns):
  ax = axs[i // 4, i % 4]
  sns.histplot(data=dfComplete.loc[:, dfComplete.columns != 'datetime'], x=coluna, ax=ax

  ax.set_title(f'Distribuição de {coluna}')
  ax.set_xlabel(f'{coluna}')
  ax.set_ylabel('Contagem')
plt.tight_layout()
plt.show()
```

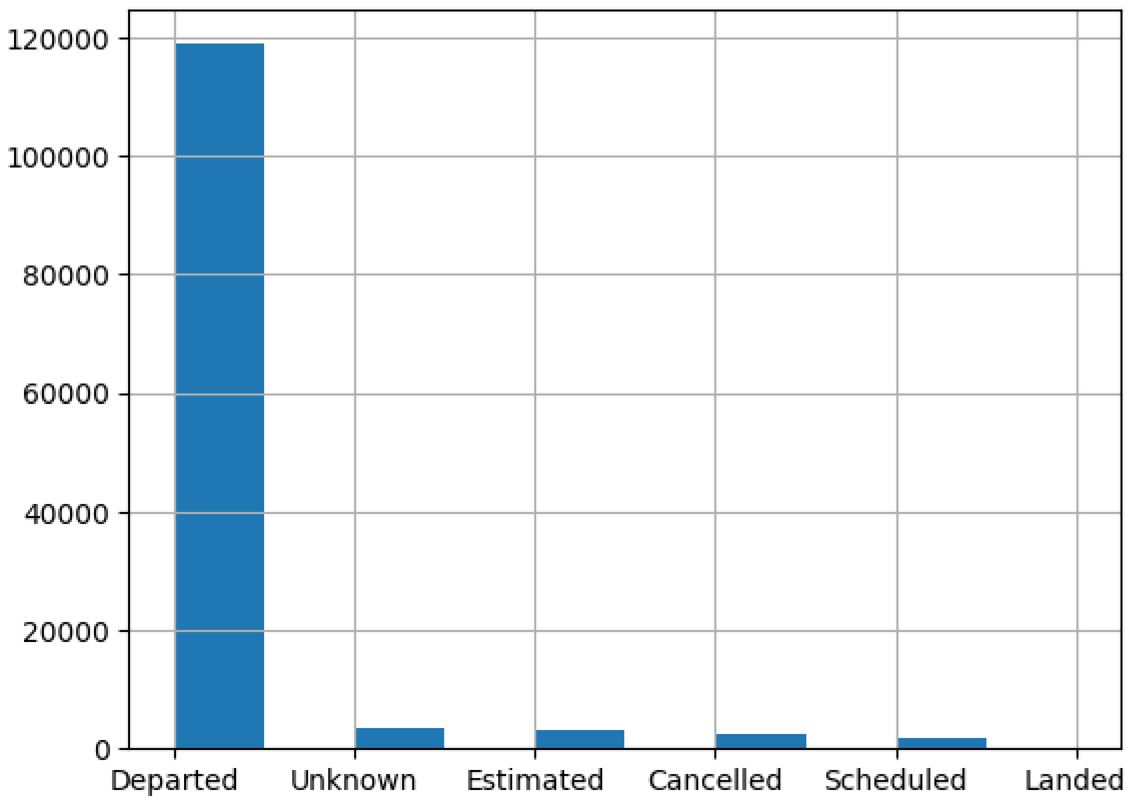




Através da análise da distribuição das variáveis, foi possível identificar algumas variáveis com valores únicos. Essas variáveis não são relevantes para a análise, pois não agregam informação adicional. Além disso, observa-se que a maioria das variáveis apresenta uma distribuição próxima à normal. Essa característica é vantajosa, pois facilita a aplicação de métodos estatísticos, que frequentemente assumem a normalidade dos dados para gerar resultados mais precisos e interpretáveis. A combinação de variáveis com distribuições normais e a ausência de dados incompletos ou categóricos excessivos contribui para a robustez e a eficácia das análises subsequentes.

```
In [ ]: dfComplete['Status'] = dfComplete['Status'].apply(lambda x: re.sub(r'\s*\d{2}:\d{2}', ''
        dfComplete['Status'].hist()
```

```
Out[ ]: <Axes: >
```



**Ajustando a variável alvo:**

- Departed: O voo já partiu do aeroporto de origem.
- Unknown: O status do voo não está disponível ou não pode ser determinado.
- Estimated: A hora de partida ou chegada foi ajustada com base em informações mais recentes, mas o voo ainda não partiu ou chegou.
- Cancelled: O voo foi cancelado e não ocorrerá.
- Scheduled: O voo está programado para partir ou chegar no horário previsto.
- Landed: O voo já pousou no aeroporto de destino.

---

- Remover: 'Unknown', nan (informações não úteis)
- Atrasados ou cancelados: 'Cancelled', 'Estimated' (problemas potenciais)
- Voo sem problemas: 'Departed', 'Scheduled', 'Landed' (dentro do previsto)

In [ ]:
```
dfComplete['Status'] = dfComplete['Status'].apply(lambda status: 0 if status in ['Depart
dfComplete = dfComplete[dfComplete['Status'] != -1]
dfComplete
```

Out[ ]:

| | IATA code | Destination | Flight | Airline | Status | Origin | datetime | temperature_2m | relative_humidity_2m |
|---|---|---|---|---|---|---|---|---|---|
| **0** | CWB | Curitiba | LA3286 | LATAM Airlines 2 | 0 | GRU | 2024-04-07 13:00:00 | 22.6 | 72 |
| **1** | CWB | Curitiba | DL7371 | Delta Air Lines | 0 | GRU | 2024-04-07 13:00:00 | 22.6 | 72 |
| **2** | CWB | Curitiba | QR5117 | Qatar Airways | 0 | GRU | 2024-04-07 13:00:00 | 22.6 | 72 |
| **3** | CWB | Curitiba | G31106 | Gol | 0 | CGH | 2024-04-07 13:00:00 | 22.5 | 76 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **4** | CWB | Curitiba | LA3248 | LATAM Airlines | 0 | CGH | 2024-04-07 13:00:00 | 22.5 | 76 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **129900** | VCP | Campinas | AD4027 | Azul 1 | 0 | SSA | 2024-05-07 02:00:00 | 26.7 | 74 |
| **129901** | VCP | Campinas | TP5324 | TAP Air Portugal | 0 | SSA | 2024-05-07 02:00:00 | 26.7 | 74 |
| **129902** | GIG | Rio De Janeiro | LA3673 | LATAM Airlines 2 | 0 | SSA | 2024-05-07 03:00:00 | 26.8 | 71 |
| **129903** | GIG | Rio De Janeiro | DL6322 | Delta Air Lines | 0 | SSA | 2024-05-07 03:00:00 | 26.8 | 71 |
| **129904** | GIG | Rio De Janeiro | LH4679 | Lufthansa | 0 | SSA | 2024-05-07 03:00:00 | 26.8 | 71 |

126303 rows × 37 columns

In [ ]:
```python
dfComplete.drop(columns='datetime', inplace=True)
dfComplete.to_csv('/content/dfCompleteWithTarget.csv', index=False)
```

```
<ipython-input-185-09a4cd97e4bc>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  dfComplete.drop(columns='datetime', inplace=True)
```

## Covariância

A covariância mede a tendência conjunta de duas variáveis em se desviarem de suas respectivas médias. Se a covariância é positiva, indica que as variáveis tendem a aumentar ou diminuir juntas. Se é negativa, uma variável tende a aumentar enquanto a outra diminui. A magnitude da covariância indica a força da relação linear entre as variáveis.

In [ ]:
```python
dfComplete[dfComplete.Status == 0].select_dtypes(include=['float64', 'int64']).cov()
```

Out[ ]:

| | Status | temperature_2m | relative_humidity_2m | dew_point_2m | apparent_temperature |
|---|---|---|---|---|---|
| **Status** | 0.0 | 0.000000 | 0.000000 | 0.000000 | 0.00000 |
| **temperature_2m** | 0.0 | 19.583078 | -54.268091 | 5.952900 | 22.52886 |
| **relative_humidity_2m** | 0.0 | -54.268091 | 303.761304 | 20.454970 | -46.80804 |
| **dew_point_2m** | 0.0 | 5.952900 | 20.454970 | 11.193149 | 10.73994 |
| **apparent_temperature** | 0.0 | 22.528865 | -46.808041 | 10.739940 | 28.08337 |
| **precipitation_probability** | NaN | NaN | NaN | NaN | Na |
| **precipitation** | 0.0 | 0.060066 | 1.139537 | 0.323563 | 0.16852 |
| **rain** | 0.0 | 0.060066 | 1.139537 | 0.323563 | 0.16852 |
| **showers** | NaN | NaN | NaN | NaN | Na |
| **snowfall** | 0.0 | 0.000000 | 0.000000 | 0.000000 | 0.00000 |

| | | | | | |
|---|---|---|---|---|---|
| **pressure_msl** | 0.0 | -6.147855 | 10.003034 | -3.599188 | -7.76768 |
| **cloud_cover** | 0.0 | -13.675548 | 241.674480 | 42.239714 | 0.099971 |
| **visibility** | NaN | NaN | NaN | NaN | Na |
| **wind_speed_10m** | 0.0 | 4.951074 | -14.693379 | 1.606650 | 2.94206 |
| **wind_direction_10m** | 0.0 | 69.234102 | -334.862605 | -15.576658 | 67.85558 |
| **wind_gusts_10m** | 0.0 | 17.560657 | -53.198259 | 4.829105 | 15.636861 |
| **temperature_2m_dst** | 0.0 | 8.684358 | -32.342991 | 0.808399 | 9.07715 |
| **relative_humidity_2m_dst** | 0.0 | -40.666209 | 158.925681 | -1.414410 | -41.31297 |
| **dew_point_2m_dst** | 0.0 | -1.211979 | 6.091498 | 0.493704 | -0.97624 |
| **apparent_temperature_dst** | 0.0 | 8.119588 | -29.577406 | 1.014809 | 8.69560 |
| **precipitation_probability_dst** | NaN | NaN | NaN | NaN | Na |
| **precipitation_dst** | 0.0 | 0.044646 | -0.155173 | 0.006385 | 0.04636 |
| **rain_dst** | 0.0 | 0.044831 | -0.155558 | 0.006503 | 0.04658 |
| **showers_dst** | NaN | NaN | NaN | NaN | Na |
| **snowfall_dst** | 0.0 | -0.000129 | 0.000270 | -0.000083 | -0.00015 |
| **pressure_msl_dst** | 0.0 | -0.240895 | 3.372732 | 0.664041 | 0.07081 |
| **cloud_cover_dst** | 0.0 | -8.306385 | 65.166052 | 8.059068 | -4.940957 |
| **visibility_dst** | NaN | NaN | NaN | NaN | Na |
| **wind_speed_10m_dst** | 0.0 | 4.974472 | -19.579353 | 0.112854 | 4.91408 |
| **wind_direction_10m_dst** | 0.0 | 4.186537 | -67.159574 | -15.180939 | 0.45183 |
| **wind_gusts_10m_dst** | 0.0 | 16.760810 | -65.890736 | 0.530405 | 16.93086 |

```
In [ ]: dfComplete[dfComplete.Status == 1].select_dtypes(include=['float64', 'int64']).cov()
```

Out[ ]:

| | Status | temperature_2m | relative_humidity_2m | dew_point_2m | apparent_temperature |
|---|---|---|---|---|---|
| **Status** | 0.0 | 0.000000 | 0.000000 | 0.000000 | 0.00000 |
| **temperature_2m** | 0.0 | 18.203319 | -54.838287 | 4.566947 | 20.79075 |
| **relative_humidity_2m** | 0.0 | -54.838287 | 320.064373 | 23.760765 | -46.95492 |
| **dew_point_2m** | 0.0 | 4.566947 | 23.760765 | 10.797224 | 9.146832 |
| **apparent_temperature** | 0.0 | 20.790754 | -46.954920 | 9.146832 | 25.93225 |
| **precipitation_probability** | NaN | NaN | NaN | NaN | Na |
| **precipitation** | 0.0 | -0.061842 | 2.003986 | 0.386081 | 0.09827 |
| **rain** | 0.0 | -0.061842 | 2.003986 | 0.386081 | 0.09827 |
| **showers** | NaN | NaN | NaN | NaN | Na |
| **snowfall** | 0.0 | 0.000000 | 0.000000 | 0.000000 | 0.00000 |
| **pressure_msl** | 0.0 | -5.386874 | 6.560141 | -3.883470 | -7.00752 |
| **cloud_cover** | 0.0 | -28.093917 | 308.017606 | 43.321754 | -14.188441 |
| **visibility** | NaN | NaN | NaN | NaN | Na |
| **wind_speed_10m** | 0.0 | 2.676438 | -10.252383 | 0.480672 | -0.13378 |
| **wind_direction_10m** | 0.0 | 99.678435 | -335.718124 | 16.432105 | 111.66401 |
| **wind_gusts_10m** | 0.0 | 12.682840 | -43.387054 | 2.410600 | 9.36125 |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| **temperature_2m_dst** | 0.0 | 10.471524 | -35.555843 | 1.869502 | 11.492172 |
| **relative_humidity_2m_dst** | 0.0 | -35.879967 | 141.999722 | -1.995749 | -37.200659 |
| **dew_point_2m_dst** | 0.0 | 1.360906 | -0.819790 | 1.091646 | 1.891286 |
| **apparent_temperature_dst** | 0.0 | 11.292827 | -36.883612 | 2.303568 | 12.623952 |
| **precipitation_probability_dst** | NaN | NaN | NaN | NaN | NaN |
| **precipitation_dst** | 0.0 | -0.054530 | 0.562899 | 0.071659 | -0.035168 |
| **rain_dst** | 0.0 | -0.054530 | 0.562899 | 0.071659 | -0.035168 |
| **showers_dst** | NaN | NaN | NaN | NaN | NaN |
| **snowfall_dst** | 0.0 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **pressure_msl_dst** | 0.0 | -2.128044 | 8.997159 | 0.103720 | -2.165822 |
| **cloud_cover_dst** | 0.0 | -0.032078 | 28.012850 | 6.004670 | 2.319683 |
| **visibility_dst** | NaN | NaN | NaN | NaN | NaN |
| **wind_speed_10m_dst** | 0.0 | 2.985185 | -9.838839 | 0.779844 | 3.156159 |
| **wind_direction_10m_dst** | 0.0 | 20.182586 | -97.942423 | -4.674788 | 15.387125 |
| **wind_gusts_10m_dst** | 0.0 | 12.025265 | -48.001224 | 0.714380 | 12.147119 |

## Correlação de Pearson

A correlação de Pearson mede a força e a direção da associação linear entre duas variáveis. Baseia-se nas médias e desvios padrão dos dados. Um coeficiente próximo de 1 ou -1 indica uma forte associação linear. Um coeficiente próximo de 0 indica pouca ou nenhuma associação linear. Um coeficiente positivo indica que as variáveis tendem a aumentar juntas, enquanto um coeficiente negativo indica que uma variável tende a aumentar enquanto a outra diminui. A correlação de Pearson é sensível a outliers, que podem distorcer o coeficiente de correlação.

```
In [ ]: dfComplete[dfComplete.Status == 0].select_dtypes(include=['float64', 'int64']).corr(meth
```

Out[ ]:

|  | Status | temperature_2m | relative_humidity_2m | dew_point_2m | apparent_temperature |
|---|---|---|---|---|---|
| **Status** | NaN | NaN | NaN | NaN | NaN |
| **temperature_2m** | NaN | 1.000000 | -0.703619 | 0.402080 | 0.960670 |
| **relative_humidity_2m** | NaN | -0.703619 | 1.000000 | 0.350797 | -0.506792 |
| **dew_point_2m** | NaN | 0.402080 | 0.350797 | 1.000000 | 0.605761 |
| **apparent_temperature** | NaN | 0.960670 | -0.506792 | 0.605761 | 1.000000 |
| **precipitation_probability** | NaN | NaN | NaN | NaN | NaN |
| **precipitation** | NaN | 0.025476 | 0.122715 | 0.181518 | 0.059685 |
| **rain** | NaN | 0.025476 | 0.122715 | 0.181518 | 0.059685 |
| **showers** | NaN | NaN | NaN | NaN | NaN |
| **snowfall** | NaN | NaN | NaN | NaN | NaN |
| **pressure_msl** | NaN | -0.509811 | 0.210616 | -0.394779 | -0.537890 |
| **cloud_cover** | NaN | -0.088597 | 0.397538 | 0.361959 | 0.000541 |
| **visibility** | NaN | NaN | NaN | NaN | NaN |
| **wind_speed_10m** | NaN | 0.244924 | -0.184556 | 0.105128 | 0.121535 |

| | | | | | |
|---|---|---|---|---|---|
| **wind_direction_10m** | NaN | 0.161951 | -0.198886 | -0.048195 | 0.132546 |
| **wind_gusts_10m** | NaN | 0.443173 | -0.340882 | 0.161199 | 0.329532 |
| **temperature_2m_dst** | NaN | 0.396737 | -0.375163 | 0.048849 | 0.346282 |
| **relative_humidity_2m_dst** | NaN | -0.534245 | 0.530121 | -0.024578 | -0.453220 |
| **dew_point_2m_dst** | NaN | -0.056566 | 0.072187 | 0.030478 | -0.038048 |
| **apparent_temperature_dst** | NaN | 0.278585 | -0.257667 | 0.046055 | 0.249138 |
| **precipitation_probability_dst** | NaN | NaN | NaN | NaN | NaN |
| **precipitation_dst** | NaN | 0.014420 | -0.012725 | 0.002728 | 0.012504 |
| **rain_dst** | NaN | 0.014481 | -0.012758 | 0.002779 | 0.012567 |
| **showers_dst** | NaN | NaN | NaN | NaN | NaN |
| **snowfall_dst** | NaN | -0.008707 | 0.004606 | -0.007356 | -0.008885 |
| **pressure_msl_dst** | NaN | -0.015554 | 0.055293 | 0.056712 | 0.003818 |
| **cloud_cover_dst** | NaN | -0.054515 | 0.108593 | 0.069961 | -0.027079 |
| **visibility_dst** | NaN | NaN | NaN | NaN | NaN |
| **wind_speed_10m_dst** | NaN | 0.211191 | -0.211058 | 0.006337 | 0.174216 |
| **wind_direction_10m_dst** | NaN | 0.009645 | -0.039284 | -0.046259 | 0.000869 |
| **wind_gusts_10m_dst** | NaN | 0.376738 | -0.376048 | 0.015769 | 0.317789 |

```
In [ ]: dfComplete[dfComplete.Status == 1].select_dtypes(include=['float64', 'int64']).corr(meth
```

Out[ ]:

| | Status | temperature_2m | relative_humidity_2m | dew_point_2m | apparent_temperature |
|---|---|---|---|---|---|
| **Status** | NaN | NaN | NaN | NaN | NaN |
| **temperature_2m** | NaN | 1.000000 | -0.718439 | 0.325758 | 0.956918 |
| **relative_humidity_2m** | NaN | -0.718439 | 1.000000 | 0.404190 | -0.515398 |
| **dew_point_2m** | NaN | 0.325758 | 0.404190 | 1.000000 | 0.546631 |
| **apparent_temperature** | NaN | 0.956918 | -0.515398 | 0.546631 | 1.000000 |
| **precipitation_probability** | NaN | NaN | NaN | NaN | NaN |
| **precipitation** | NaN | -0.021130 | 0.163292 | 0.171282 | 0.028131 |
| **rain** | NaN | -0.021130 | 0.163292 | 0.171282 | 0.028131 |
| **showers** | NaN | NaN | NaN | NaN | NaN |
| **snowfall** | NaN | NaN | NaN | NaN | NaN |
| **pressure_msl** | NaN | -0.434644 | 0.126231 | -0.406851 | -0.473714 |
| **cloud_cover** | NaN | -0.178871 | 0.467693 | 0.358141 | -0.075686 |
| **visibility** | NaN | NaN | NaN | NaN | NaN |
| **wind_speed_10m** | NaN | 0.135480 | -0.123766 | 0.031593 | -0.005674 |
| **wind_direction_10m** | NaN | 0.227224 | -0.182509 | 0.048637 | 0.213266 |
| **wind_gusts_10m** | NaN | 0.345991 | -0.282270 | 0.085387 | 0.213962 |
| **temperature_2m_dst** | NaN | 0.528698 | -0.428120 | 0.122558 | 0.486133 |
| **relative_humidity_2m_dst** | NaN | -0.461202 | 0.435294 | -0.033309 | -0.400631 |
| **dew_point_2m_dst** | NaN | 0.078633 | -0.011296 | 0.081899 | 0.091556 |
| **apparent_temperature_dst** | NaN | 0.458776 | -0.357345 | 0.121512 | 0.429683 |

| | | | | | |
|---|---|---|---|---|---|
| **precipitation_probability_dst** | NaN | NaN | NaN | NaN | NaN |
| **precipitation_dst** | NaN | -0.013995 | 0.034452 | 0.023879 | -0.007562 |
| **rain_dst** | NaN | -0.013995 | 0.034452 | 0.023879 | -0.007562 |
| **showers_dst** | NaN | NaN | NaN | NaN | NaN |
| **snowfall_dst** | NaN | NaN | NaN | NaN | NaN |
| **pressure_msl_dst** | NaN | -0.163006 | 0.164355 | 0.010316 | -0.138995 |
| **cloud_cover_dst** | NaN | -0.000207 | 0.043119 | 0.050322 | 0.012544 |
| **visibility_dst** | NaN | NaN | NaN | NaN | NaN |
| **wind_speed_10m_dst** | NaN | 0.137831 | -0.108336 | 0.046752 | 0.122092 |
| **wind_direction_10m_dst** | NaN | 0.045442 | -0.052591 | -0.013667 | 0.029026 |
| **wind_gusts_10m_dst** | NaN | 0.305783 | -0.291091 | 0.023587 | 0.258790 |

## Correlação de Spearman

A correlação de Spearman mede a força e a direção da associação monotônica (não necessariamente linear) entre duas variáveis. Baseia-se nas posições (ranks) dos dados. Um coeficiente próximo de 1 ou -1 indica uma forte associação monotônica. Um coeficiente próximo de 0 indica pouca ou nenhuma associação monotônica. Como se baseia em ranks, a correlação de Spearman é menos sensível a outliers do que a correlação de Pearson. É útil para identificar relações monotônicas não lineares que a correlação de Pearson poderia não capturar.

```
In [ ]: dfComplete[dfComplete.Status == 0].select_dtypes(include=['float64', 'int64']).corr(meth
```

Out[ ]:

| | Status | temperature_2m | relative_humidity_2m | dew_point_2m | apparent_temperature |
|---|---|---|---|---|---|
| **Status** | NaN | NaN | NaN | NaN | NaN |
| **temperature_2m** | NaN | 1.000000 | -0.694402 | 0.395240 | 0.961384 |
| **relative_humidity_2m** | NaN | -0.694402 | 1.000000 | 0.306013 | -0.509776 |
| **dew_point_2m** | NaN | 0.395240 | 0.306013 | 1.000000 | 0.585277 |
| **apparent_temperature** | NaN | 0.961384 | -0.509776 | 0.585277 | 1.000000 |
| **precipitation_probability** | NaN | NaN | NaN | NaN | NaN |
| **precipitation** | NaN | 0.156737 | 0.170691 | 0.405840 | 0.241721 |
| **rain** | NaN | 0.156737 | 0.170691 | 0.405840 | 0.241721 |
| **showers** | NaN | NaN | NaN | NaN | NaN |
| **snowfall** | NaN | NaN | NaN | NaN | NaN |
| **pressure_msl** | NaN | -0.516620 | 0.239719 | -0.349468 | -0.541640 |
| **cloud_cover** | NaN | -0.006104 | 0.387261 | 0.442065 | 0.081834 |
| **visibility** | NaN | NaN | NaN | NaN | NaN |
| **wind_speed_10m** | NaN | 0.262864 | -0.221756 | 0.087370 | 0.137870 |
| **wind_direction_10m** | NaN | 0.202088 | -0.148465 | 0.048600 | 0.199250 |
| **wind_gusts_10m** | NaN | 0.473164 | -0.379712 | 0.141132 | 0.355329 |
| **temperature_2m_dst** | NaN | 0.446507 | -0.423280 | 0.047154 | 0.383349 |
| **relative_humidity_2m_dst** | NaN | -0.538499 | 0.535836 | -0.023669 | -0.454179 |

| | | | | | |
|---|---|---|---|---|---|
| **dew_point_2m_dst** | NaN | -0.054960 | 0.056692 | 0.038124 | -0.042325 |
| **apparent_temperature_dst** | NaN | 0.303786 | -0.284928 | 0.042959 | 0.264032 |
| **precipitation_probability_dst** | NaN | NaN | NaN | NaN | NaN |
| **precipitation_dst** | NaN | 0.043548 | -0.031105 | 0.035691 | 0.042862 |
| **rain_dst** | NaN | 0.043548 | -0.031102 | 0.035699 | 0.042864 |
| **showers_dst** | NaN | NaN | NaN | NaN | NaN |
| **snowfall_dst** | NaN | -0.009095 | 0.006751 | -0.008724 | -0.009323 |
| **pressure_msl_dst** | NaN | -0.005104 | 0.037455 | 0.041289 | 0.011644 |
| **cloud_cover_dst** | NaN | -0.044333 | 0.099049 | 0.089609 | -0.019684 |
| **visibility_dst** | NaN | NaN | NaN | NaN | NaN |
| **wind_speed_10m_dst** | NaN | 0.226980 | -0.225201 | 0.014298 | 0.188456 |
| **wind_direction_10m_dst** | NaN | -0.010796 | -0.019582 | -0.057421 | -0.018775 |
| **wind_gusts_10m_dst** | NaN | 0.406360 | -0.397469 | 0.030412 | 0.343894 |

In [ ]:
```
dfComplete[dfComplete.Status == 1].select_dtypes(include=['float64', 'int64']).corr(meth
```

Out[ ]:

| | Status | temperature_2m | relative_humidity_2m | dew_point_2m | apparent_temperature |
|---|---|---|---|---|---|
| **Status** | NaN | NaN | NaN | NaN | NaN |
| **temperature_2m** | NaN | 1.000000 | -0.698293 | 0.350649 | 0.960414 |
| **relative_humidity_2m** | NaN | -0.698293 | 1.000000 | 0.332077 | -0.513740 |
| **dew_point_2m** | NaN | 0.350649 | 0.332077 | 1.000000 | 0.544484 |
| **apparent_temperature** | NaN | 0.960414 | -0.513740 | 0.544484 | 1.000000 |
| **precipitation_probability** | NaN | NaN | NaN | NaN | NaN |
| **precipitation** | NaN | 0.103243 | 0.197560 | 0.394189 | 0.180487 |
| **rain** | NaN | 0.103243 | 0.197560 | 0.394189 | 0.180487 |
| **showers** | NaN | NaN | NaN | NaN | NaN |
| **snowfall** | NaN | NaN | NaN | NaN | NaN |
| **pressure_msl** | NaN | -0.443776 | 0.162424 | -0.421964 | -0.481252 |
| **cloud_cover** | NaN | -0.127282 | 0.500776 | 0.444989 | -0.022314 |
| **visibility** | NaN | NaN | NaN | NaN | NaN |
| **wind_speed_10m** | NaN | 0.147133 | -0.179152 | 0.016547 | 0.006970 |
| **wind_direction_10m** | NaN | 0.230727 | -0.128188 | 0.108511 | 0.232057 |
| **wind_gusts_10m** | NaN | 0.366627 | -0.327607 | 0.078238 | 0.232913 |
| **temperature_2m_dst** | NaN | 0.548092 | -0.461050 | 0.118481 | 0.500192 |
| **relative_humidity_2m_dst** | NaN | -0.466199 | 0.469864 | -0.025278 | -0.399666 |
| **dew_point_2m_dst** | NaN | 0.097427 | -0.021080 | 0.079437 | 0.105334 |
| **apparent_temperature_dst** | NaN | 0.477574 | -0.376581 | 0.118133 | 0.442807 |
| **precipitation_probability_dst** | NaN | NaN | NaN | NaN | NaN |
| **precipitation_dst** | NaN | 0.064651 | -0.042270 | 0.035646 | 0.057800 |
| **rain_dst** | NaN | 0.064651 | -0.042270 | 0.035646 | 0.057800 |
| **showers_dst** | NaN | NaN | NaN | NaN | NaN |

| | | | | | |
|---|---|---|---|---|---|
| **snowfall_dst** | NaN | NaN | NaN | NaN | NaN |
| **pressure_msl_dst** | NaN | -0.152840 | 0.153711 | 0.006409 | -0.137333 |
| **cloud_cover_dst** | NaN | 0.004779 | 0.040465 | 0.043710 | 0.018097 |
| **visibility_dst** | NaN | NaN | NaN | NaN | NaN |
| **wind_speed_10m_dst** | NaN | 0.134193 | -0.122883 | 0.030397 | 0.117553 |
| **wind_direction_10m_dst** | NaN | 0.047622 | -0.043690 | -0.013036 | 0.035586 |
| **wind_gusts_10m_dst** | NaN | 0.329514 | -0.331560 | 0.009125 | 0.278104 |

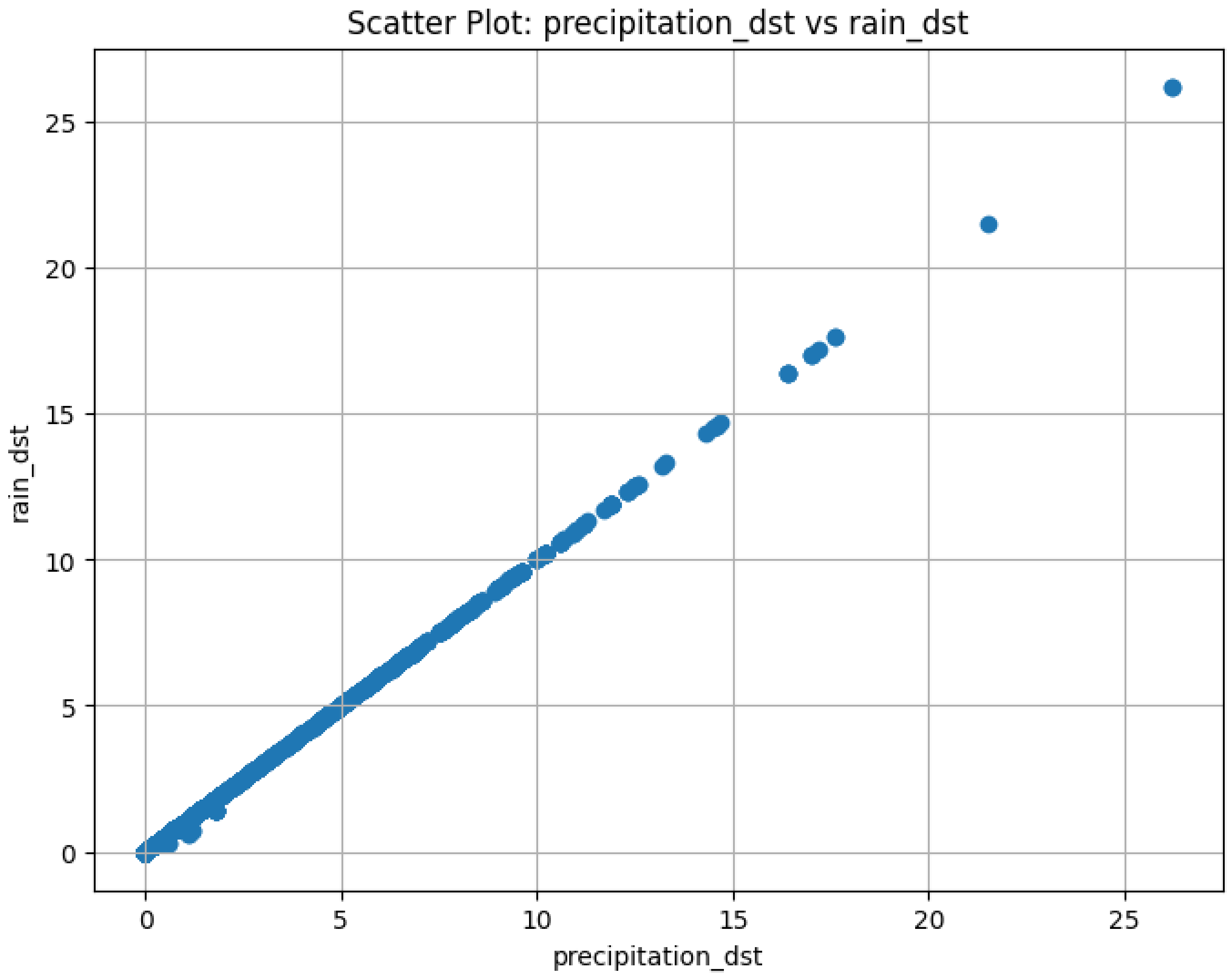
## Comportamento de Pares de Variáveis Altamente Relacionadas

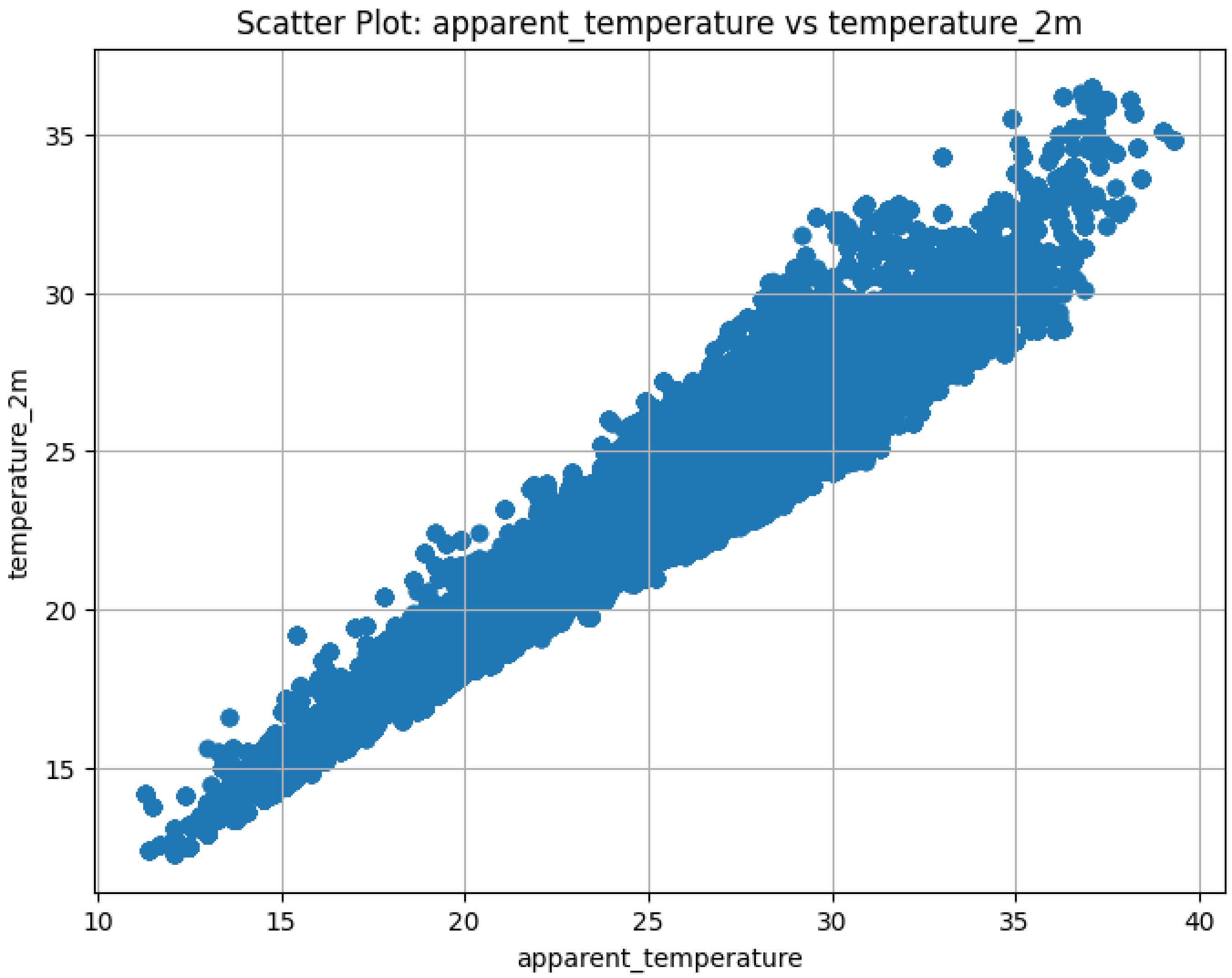
In [ ]:
```
corr_spearman = dfComplete.select_dtypes(include=['float64', 'int64']).corr(method='spea
high_corr_pairs = corr_spearman.abs().unstack().sort_values(ascending=False)
high_corr_pairs = high_corr_pairs[(high_corr_pairs > 0.85) & (high_corr_pairs < 1)].rese
high_corr_pairs = high_corr_pairs[high_corr_pairs.index % 2 == 0].reset_index().drop(col
high_corr_pairs
```

Out[ ]:

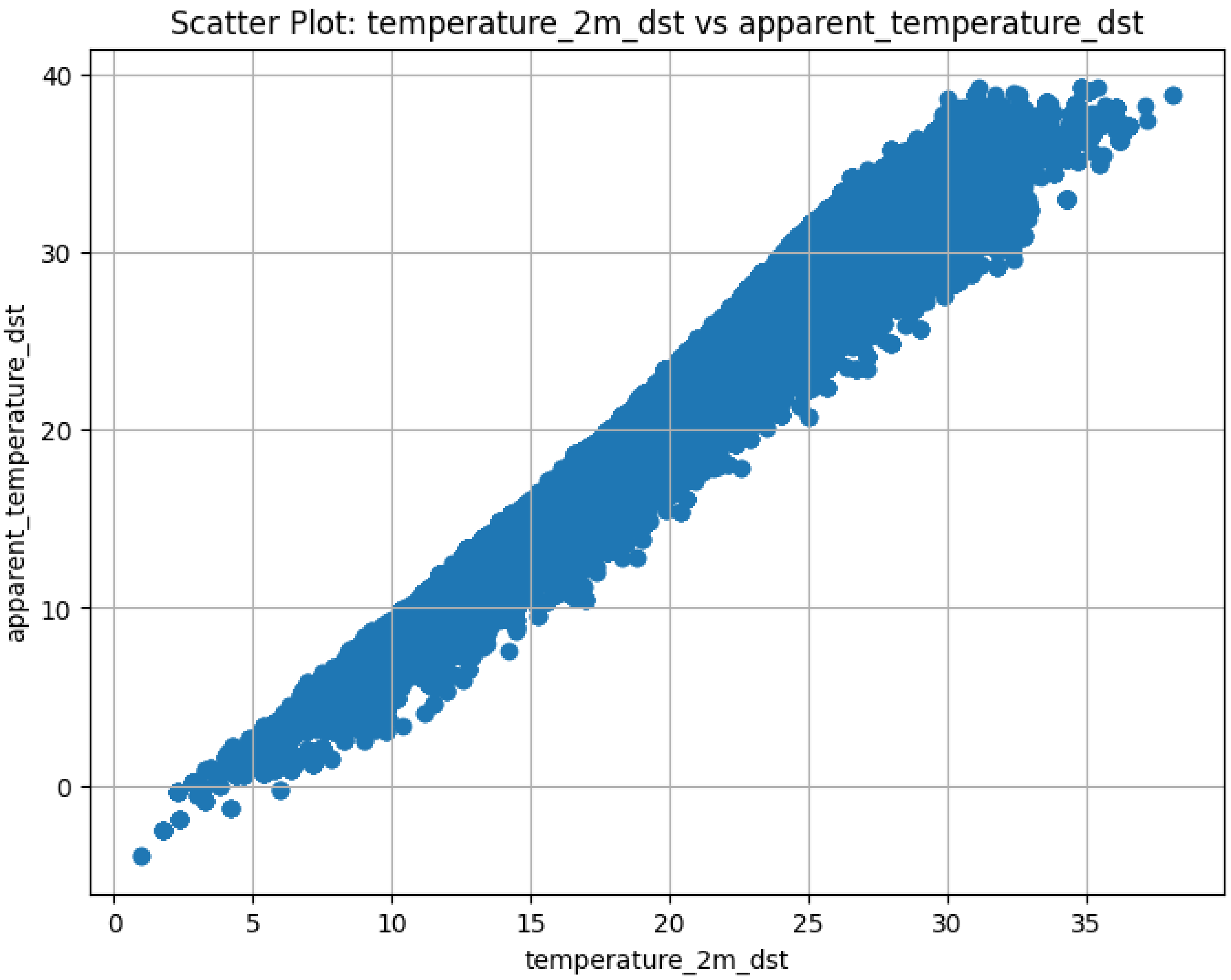
| | level_0 | level_1 | 0 |
|---|---|---|---|
| **0** | precipitation_dst | rain_dst | 1.000000 |
| **1** | apparent_temperature | temperature_2m | 0.961459 |
| **2** | temperature_2m_dst | apparent_temperature_dst | 0.938254 |
| **3** | wind_gusts_10m_dst | wind_speed_10m_dst | 0.894117 |
| **4** | wind_speed_10m | wind_gusts_10m | 0.889226 |

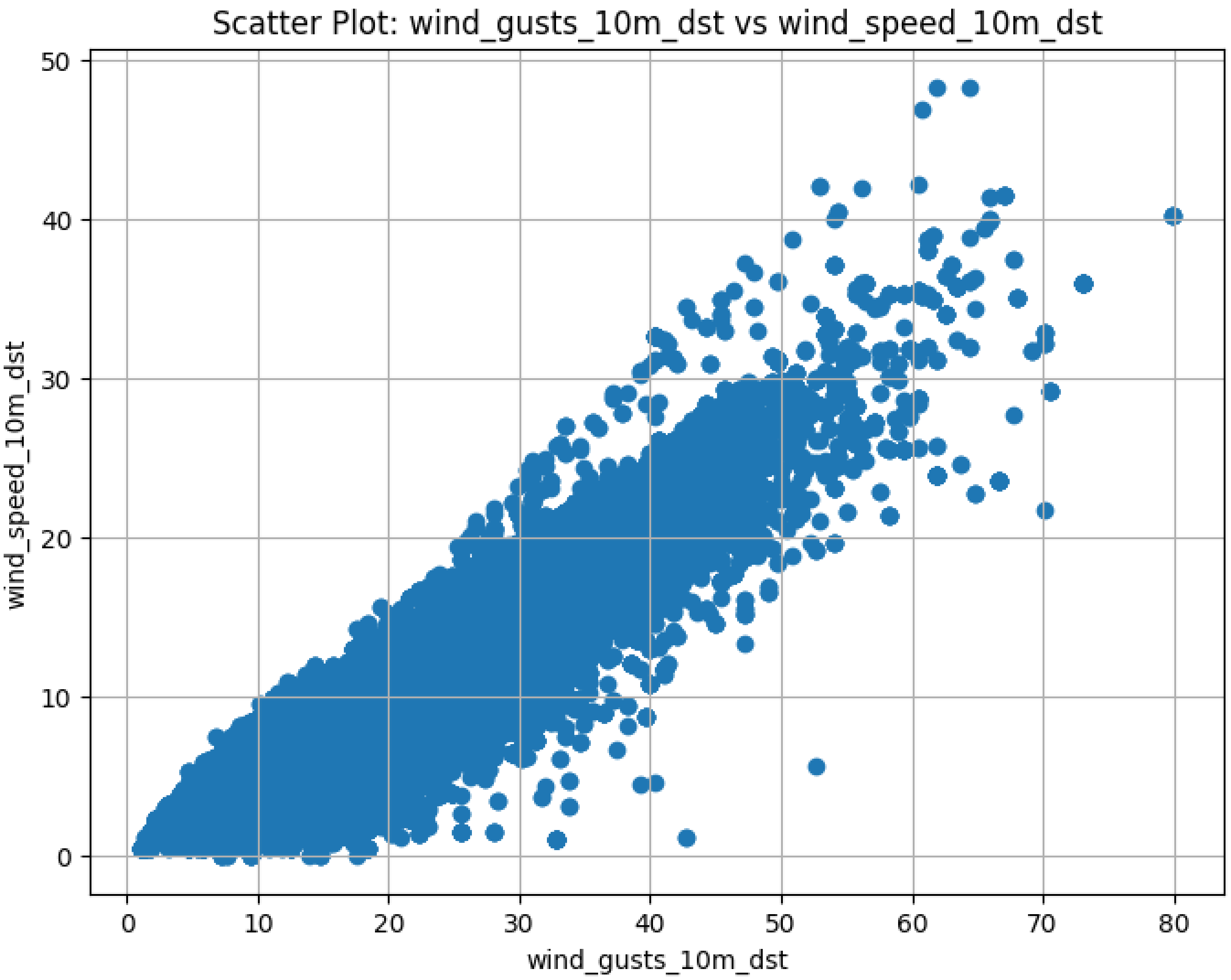
In [ ]:
```
def scatter_plot(pair):
  var1, var2, _ = pair
  plt.figure(figsize=(8, 6))
  plt.scatter(dfComplete[var1], dfComplete[var2])
  plt.xlabel(var1)
  plt.ylabel(var2)
  plt.title(f"Scatter Plot: {var1} vs {var2}")
  plt.grid(True)
  plt.show()

high_corr_pairs.apply(scatter_plot, axis=1)
```

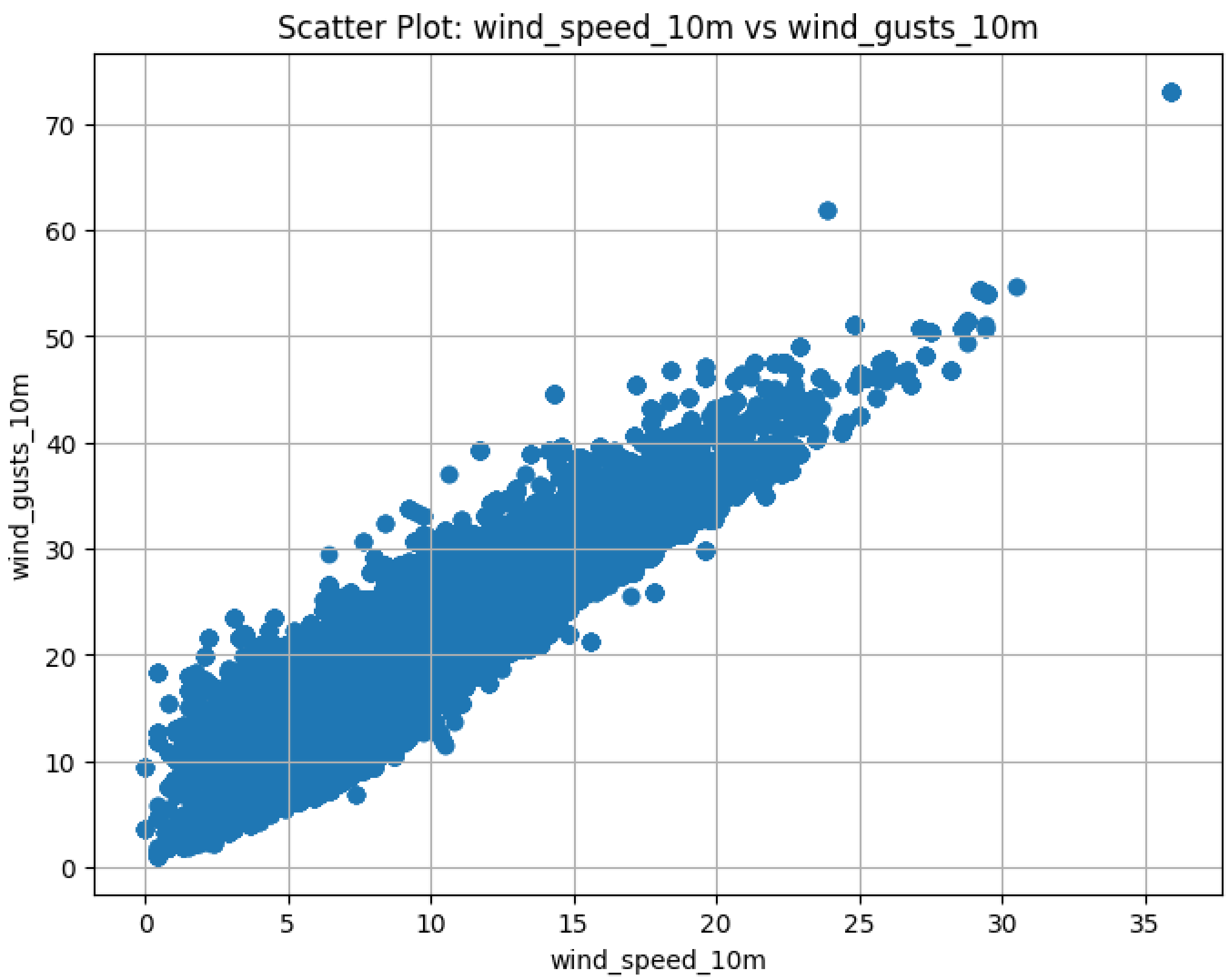

Scatter Plot: precipitation_dst vs rain_dst
rain_dst
precipitation_dst
0
5
10
15
20
25



Scatter Plot: apparent_temperature vs temperature_2m
temperature_2m
apparent_temperature
10
15
20
25
30
35
40
15
20
25
30
35


Scatter Plot: temperature_2m_dst vs apparent_temperature_dst
apparent_temperature_dst
temperature_2m_dst
0
5
10
15
20
25
30
35
40


Scatter Plot: wind_gusts_10m_dst vs wind_speed_10m_dst
wind_speed_10m_dst
wind_gusts_10m_dst
0
10
20
30
40
50
60
70
80



```
Out[ ]: 0    None
        1    None
        2    None
        3    None
        4    None
        dtype: object
```

## Pré-Processamento dos Dados

- Definição de tipos
- Tratamento de dados ausentes
- Normalização e discretização
- Limpeza de dados (univariado, bivariado e multivariado)

```
In [ ]: dfComplete = pd.read_csv('/content/dfCompleteWithTarget.csv')
        dfComplete
```

Out[ ]:

| | IATA code | Destination | Flight | Airline | Status | Origin | temperature_2m | relative_humidity_2m | dew_poin |
|---|---|---|---|---|---|---|---|---|---|
| **0** | CWB | Curitiba | LA3286 | LATAM Airlines 2 | 0 | GRU | 22.6 | 72 | |
| **1** | CWB | Curitiba | DL7371 | Delta Air Lines | 0 | GRU | 22.6 | 72 | |
| **2** | CWB | Curitiba | QR5117 | Qatar Airways | 0 | GRU | 22.6 | 72 | |
| **3** | CWB | Curitiba | G31106 | Gol | 0 | CGH | 22.5 | 76 | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **4** | CWB | Curitiba | LA3248 | LATAM Airlines | 0 | CGH | 22.5 | 76 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **126298** | VCP | Campinas | AD4027 | Azul 1 | 0 | SSA | 26.7 | 74 |
| **126299** | VCP | Campinas | TP5324 | TAP Air Portugal | 0 | SSA | 26.7 | 74 |
| **126300** | GIG | Rio De Janeiro | LA3673 | LATAM Airlines 2 | 0 | SSA | 26.8 | 71 |
| **126301** | GIG | Rio De Janeiro | DL6322 | Delta Air Lines | 0 | SSA | 26.8 | 71 |
| **126302** | GIG | Rio De Janeiro | LH4679 | Lufthansa | 0 | SSA | 26.8 | 71 |

126303 rows × 36 columns

## Informação Redundante

Linhas do conjunto de dados que possui apenas valores repetidos não adicionam nenhuma variabilidade ou informação útil à análise. A remoção dessas linhas diminui a complexidade do modelo, facilita a visualização e análise, além de reduzir o tempo de processamento.

In [ ]:
```
print(dfComplete.shape)
dfComplete = dfComplete.drop_duplicates()
dfComplete.shape
```

```
(126303, 36)
```
Out[ ]:
```
(102190, 36)
```

Identificando colunas vazias (todos os valores são nulos) ou com apenas um valor (não acrescentam informação à análise).

In [ ]:
```
colunas_vazias = dfComplete.columns[dfComplete.isna().all()].tolist()
colunas_valores_iguais = dfComplete.columns[dfComplete.apply(lambda x: x.nunique()) == 1
colunas_vazias += colunas_valores_iguais
print(colunas_vazias)

dfComplete = dfComplete.drop(columns=colunas_vazias)
```

```
['precipitation_probability', 'showers', 'visibility', 'precipitation_probability_dst',
'showers_dst', 'visibility_dst', 'snowfall']
```

## Definição de Tipos

In [ ]:
```
dfComplete.dtypes
```

Out[ ]:
```
IATA code                    object
Destination                  object
Flight                       object
Airline                      object
Status                        int64
Origin                       object
temperature_2m              float64
relative_humidity_2m          int64
dew_point_2m                float64
apparent_temperature        float64
```

```
precipitation                float64
rain                         float64
pressure_msl                 float64
cloud_cover                    int64
wind_speed_10m               float64
wind_direction_10m             int64
wind_gusts_10m               float64
temperature_2m_dst           float64
relative_humidity_2m_dst       int64
dew_point_2m_dst             float64
apparent_temperature_dst     float64
precipitation_dst            float64
rain_dst                     float64
snowfall_dst                 float64
pressure_msl_dst             float64
cloud_cover_dst                int64
wind_speed_10m_dst           float64
wind_direction_10m_dst         int64
wind_gusts_10m_dst           float64
dtype: object
```

Todas as informações contidas na tabela de clima são de natureza numérica, representadas como inteiros ou float. Portanto, apenas as colunas presentes no conjunto de dados de voos requerem avaliação individual por serem dados categóricos.

In [ ]:
```
dfComplete['IATA code'] = dfComplete['IATA code'].astype('category')
dfComplete['Destination'] = dfComplete['Destination'].astype('category')
dfComplete['Flight'] = dfComplete['Flight'].astype('category')
dfComplete['Airline'] = dfComplete['Airline'].astype('category')
dfComplete['Origin'] = dfComplete['Origin'].astype('category')

print('IATA code:', dfComplete['IATA code'].cat.categories)
print('Destination:', dfComplete['Destination'].cat.categories)
print('Flight:', dfComplete['Flight'].cat.categories)
print('Airline:', dfComplete['Airline'].cat.categories)
print('Origin:', dfComplete['Origin'].cat.categories)
```

```
IATA code: Index(['AAX', 'ADD', 'AEP', 'AJU', 'ALQ', 'AMS', 'ARU', 'ASU', 'ATL', 'ATM',
       ...
       'URG', 'VAG', 'VCP', 'VDC', 'VIX', 'VVI', 'XAP', 'YUL', 'YYZ', 'ZRH'],
      dtype='object', length=162)
Destination: Index(['Addis Ababa', 'Alegrete', 'Amsterdam', 'Aracaju', 'Aracatuba',
       'Araguaina', 'Araxa', 'Asuncion', 'Atlanta', 'Bage',
       ...
       'Toronto', 'Tres Lagoas', 'Uberaba', 'Uberlandia', 'Una', 'Uruguaiana',
       'Varginha', 'Vitoria', 'Vitoria Da Conquista', 'Zurich'],
      dtype='object', length=157)
Flight: Index(['2Z2201', '2Z2203', '2Z2205', '2Z2207', '2Z2209', '2Z2210', '2Z2213',
       '2Z2215', '2Z2216', '2Z2217',
       ...
       'VS7818', 'VS7819', 'VS7822', 'VS7823', 'VS7831', 'W8926', 'WB1225',
       'WB1342', 'WD5800', 'WD5801'],
      dtype='object', length=5293)
Airline: Index(['ANA', 'ASKY', 'Aero FlightOps UK', 'Aerolineas Argentinas',
       'Aerolineas Argentinas  1', 'Aerolineas Argentinas  2',
       'Aerolineas Argentinas  3', 'Aerolineas Argentinas  4',
       'Aerolineas Argentinas  5', 'Aeromexico',
       ...
       'Turkish Airlines', 'Turkish Airlines  2', 'Turkish Airlines  3',
       'Turkish Airlines  4', 'United Airlines', 'United Airlines  1',
       'Virgin Atlantic', 'VoePass', 'VoePass  1', 'VoePass  2'],
      dtype='object', length=154)
Origin: Index(['BSB', 'CGH', 'CNF', 'GIG', 'GRU', 'POA', 'REC', 'SDU', 'SSA', 'VCP'], dt
ype='object')
```

Com base na análise do conjunto de dados, identificamos que há 105.038 registros. A categoria com a maior variedade contém 5.322 tipos distintos, indicando sua potencial relevância para os resultados da análise. Uma variedade muito alta, aproximando-se do número de registros indicaria uma coluna com comportamento identificadora, não sendo útil a análise.

O método cat.codes é mais apropriado quando existe uma ordem intrínseca nos dados categóricos. Entretanto, para simplificação e para garantir a uniformidade dos tipos de dados como numéricos, será empregado esse método para as variáveis categóricas.

In [ ]:
```python
dfComplete['IATA code'] = dfComplete['IATA code'].cat.codes
dfComplete['Destination'] = dfComplete['Destination'].cat.codes
dfComplete['Flight'] = dfComplete['Flight'].cat.codes
dfComplete['Airline'] = dfComplete['Airline'].cat.codes
dfComplete['Origin'] = dfComplete['Origin'].cat.codes
```

In [ ]:
```python
dfComplete.dtypes
```

Out[ ]:
```
IATA code                     int16
Destination                   int16
Flight                        int16
Airline                       int16
Status                        int64
Origin                         int8
temperature_2m              float64
relative_humidity_2m          int64
dew_point_2m                float64
apparent_temperature        float64
precipitation               float64
rain                        float64
pressure_msl                float64
cloud_cover                   int64
wind_speed_10m              float64
wind_direction_10m            int64
wind_gusts_10m              float64
temperature_2m_dst          float64
relative_humidity_2m_dst      int64
dew_point_2m_dst            float64
apparent_temperature_dst    float64
precipitation_dst           float64
rain_dst                    float64
snowfall_dst                float64
pressure_msl_dst            float64
cloud_cover_dst               int64
wind_speed_10m_dst          float64
wind_direction_10m_dst        int64
wind_gusts_10m_dst          float64
dtype: object
```

## Visualização inicial

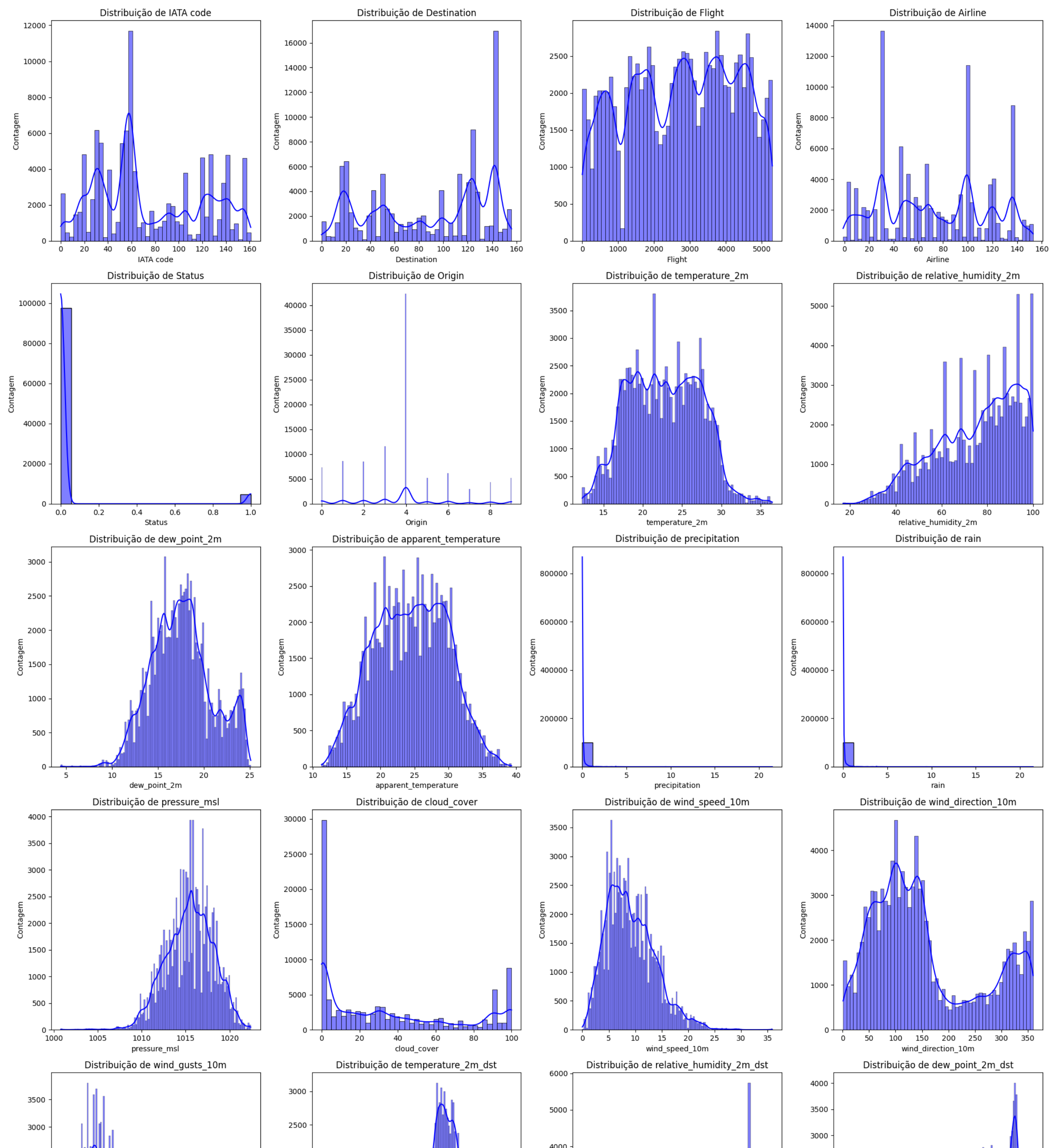
In [ ]:
```python
dfComplete.describe()
```

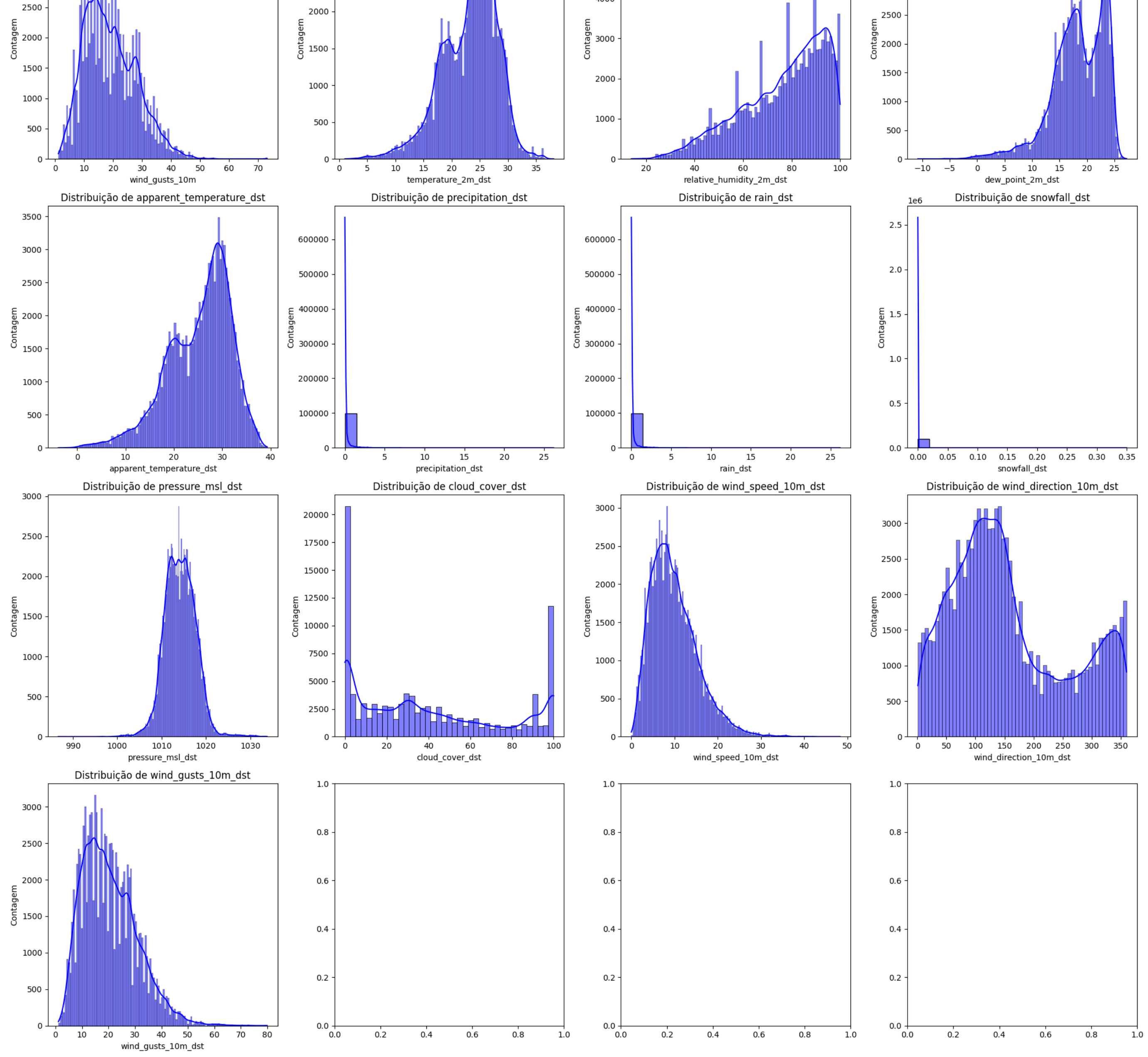
Out[ ]:

| | IATA code | Destination | Flight | Airline | Status | Origin | temperature |
|---|---|---|---|---|---|---|---|
| **count** | 102190.000000 | 102190.000000 | 102190.000000 | 102190.000000 | 102190.000000 | 102190.000000 | 102190.00 |
| **mean** | 74.581270 | 87.714855 | 2744.192308 | 73.160936 | 0.045220 | 3.863235 | 22.74 |
| **std** | 44.381904 | 47.754533 | 1486.453395 | 42.188062 | 0.207787 | 2.201285 | 4.49 |
| **min** | 0.000000 | 0.000000 | 0.000000 | -1.000000 | 0.000000 | 0.000000 | 12.30 |
| **25%** | 35.000000 | 41.000000 | 1508.000000 | 31.000000 | 0.000000 | 3.000000 | 19.10 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **50%** | 59.000000 | 99.000000 | 2824.000000 | 69.000000 | 0.000000 | 4.000000 | 22.70 |
| **75%** | 120.000000 | 129.000000 | 3995.750000 | 102.000000 | 0.000000 | 4.000000 | 26.40 |
| **max** | 161.000000 | 156.000000 | 5292.000000 | 153.000000 | 1.000000 | 9.000000 | 36.50 |

In [ ]:
```
fig, axs = plt.subplots(8, 4, figsize=(20, 5 * 8))
for i, coluna in enumerate(dfComplete.columns):
  ax = axs[i // 4, i % 4]

  sns.histplot(data=dfComplete, x=coluna, ax=ax, color='blue', alpha=0.5, kde=True)

  ax.set_title(f'Distribuição de {coluna}')
  ax.set_xlabel(f'{coluna}')
  ax.set_ylabel('Contagem')
plt.tight_layout()
plt.show()
```





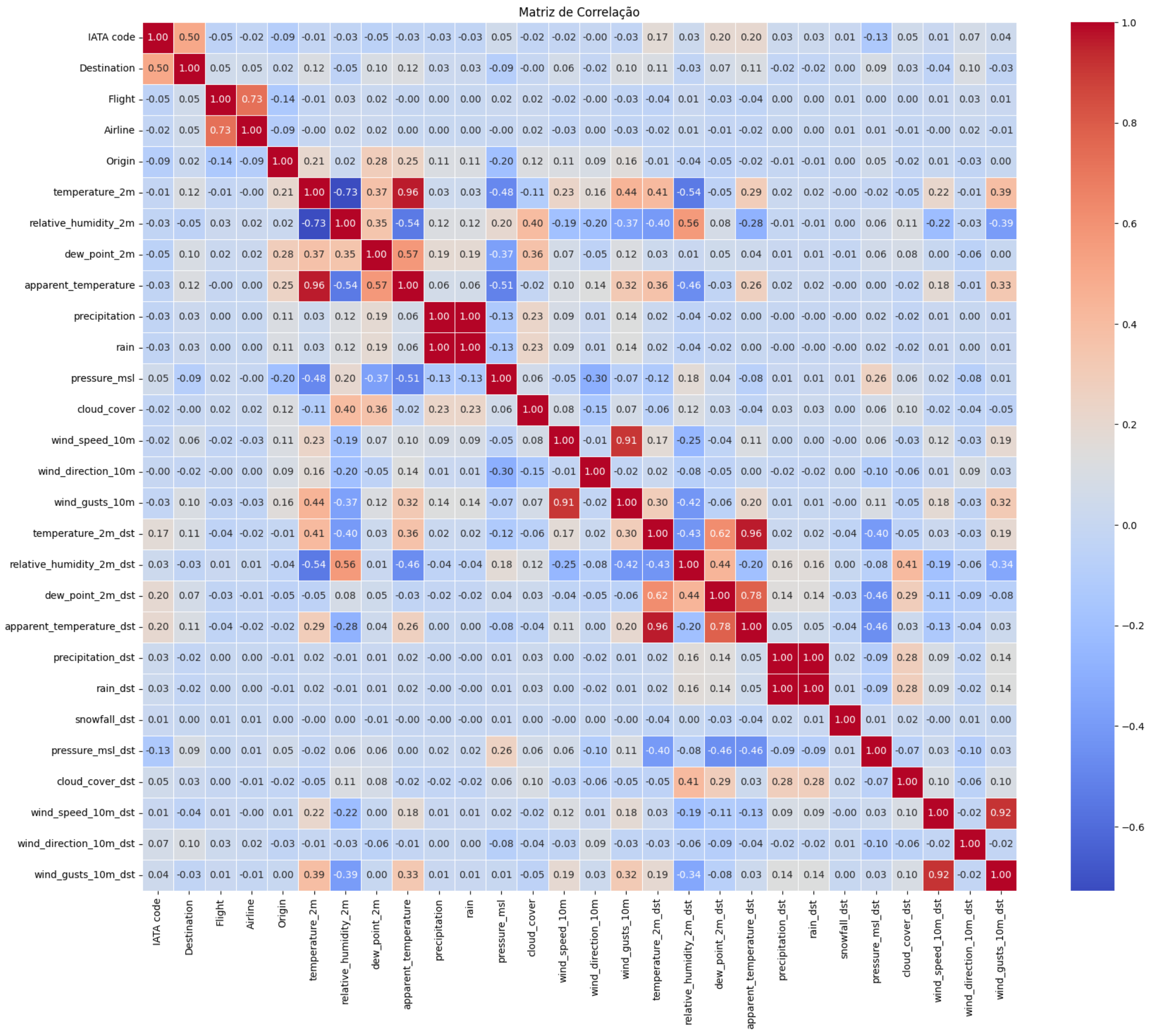
Na primeira análise, observa-se que a maioria das variáveis segue uma distribuição normal, o que permite a aplicação de diversos métodos estatísticos e analíticos. No entanto, há um evidente desbalanceamento nos dados, particularmente na variável 'Status de voo', onde uma classe é significativamente mais predominante em relação às demais. Este desbalanceamento pode influenciar os resultados das análises e previsões, indicando a necessidade de abordar esse problema em etapas subsequentes para garantir a precisão e a robustez das conclusões.

## Correlação entre Variáveis

A análise de correlação entre variáveis é uma etapa essencial no entendimento da estrutura de uma base de dados. Além de fornecer insights sobre o comportamento dos dados, essa análise permite identificar variáveis fortemente relacionadas, o que pode indicar redundância no conjunto de dados. A presença de variáveis altamente correlacionadas aumenta a dimensionalidade do modelo sem necessariamente agregar novas informações relevantes, o que pode prejudicar a eficiência computacional e a interpretabilidade do modelo.

Dado o número substancial de colunas em nosso conjunto de dados, é prudente conduzir a análise de correlação por tipo de variável. Isso permite uma visualização mais organizada e compreensível, facilitando a identificação de padrões e relações entre as variáveis. Ao segmentar a análise por tipo de variável, como numérico ou categórico, podemos explorar de forma mais eficaz as relações entre os diferentes tipos de dados presentes na base de dados.

```
In [ ]: correlation_matrix = dfComplete.loc[:, ~dfComplete.columns.isin(['Status'])].corr()

        plt.figure(figsize=(20, 16))
        sns.heatmap(correlation_matrix , annot=True, fmt=".2f", cmap='coolwarm', linewidths=0.5)
        plt.title('Matriz de Correlação')
        plt.show()
```



```
In [ ]: high_correlation_pairs = []
        cols = correlation_matrix.columns

        for i in range(len(cols)):
          for j in range(i + 1, len(cols)):
            if abs(correlation_matrix.iloc[i, j]) > 0.85:
              high_correlation_pairs.append((cols[i], cols[j], correlation_matrix.iloc[i, j]))

        print("Pares de colunas com correlação maior que 0.85:")
        for col1, col2, corr in high_correlation_pairs:
          print(f"{col1} e {col2}: {corr:.2f}")
```

```
Pares de colunas com correlação maior que 0.85:
```

```
temperature_2m e apparent_temperature: 0.96
precipitation e rain: 1.00
wind_speed_10m e wind_gusts_10m: 0.91
temperature_2m_dst e apparent_temperature_dst: 0.96
precipitation_dst e rain_dst: 1.00
wind_speed_10m_dst e wind_gusts_10m_dst: 0.92
```

Essas variáveis que ultrapassaram o limiar de correlação, definido como 85%, são consideradas redundantes para o conjunto de dados, pois apresentam uma alta correlação entre si, o que pode introduzir multicolinearidade e aumentar a complexidade sem fornecer informações adicionais significativas. A identificação e remoção dessas variáveis podem ajudar a simplificar o conjunto de dados e melhorar a eficiência e interpretabilidade da análise.

Esta abordagem visa eliminar a redundância nas características, preservando a qualidade e a relevância das informações contidas no conjunto de dados. Ao reduzir o número de características altamente correlacionadas, podemos mitigar o risco de overfitting e melhorar a capacidade do modelo de generalizar para novos dados.

In [ ]:
```
columns_to_remove = set()

for col1, col2, _ in high_correlation_pairs:
  columns_to_remove.add(col2)

dfComplete = dfComplete.drop(columns=columns_to_remove)
```

## Limpeza de dados (Detecção de Outliers)

- Univariado
  - Z-Score robusto
  - Tukey
- Bivariado
  - Transformar em Univariado (Razão de uma variável pela outra)
- Multivariado
  - Elliptic Envelope
  - Distância Mahalanobis
  - Isolation Forests

Os outliers podem distorcer a distribuição dos dados e influenciar significativamente as medidas de tendência central, comprometendo a análise estatística. Portanto, é essencial identificar e remover os outliers a fim de realizar uma análise mais precisa e robusta dos dados. Para isso, são empregadas técnicas estatísticas específicas.

A média é uma medida de tendência central que pode ser significativamente influenciada por outliers. Portanto, ao comparar a média com a mediana, podemos estimar a presença de outliers.

In [ ]:
```
dfComplete.describe()
```

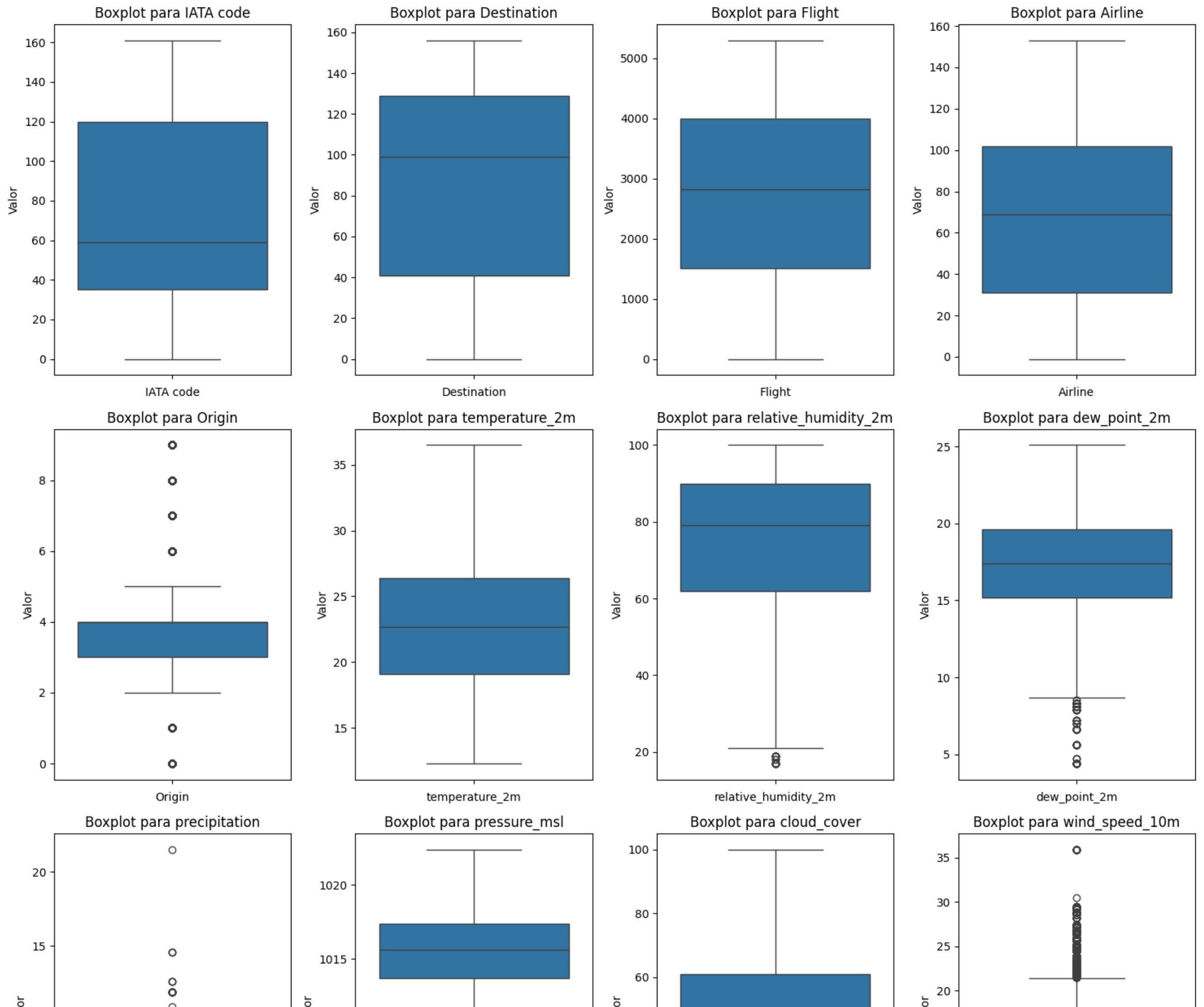
Out[ ]:

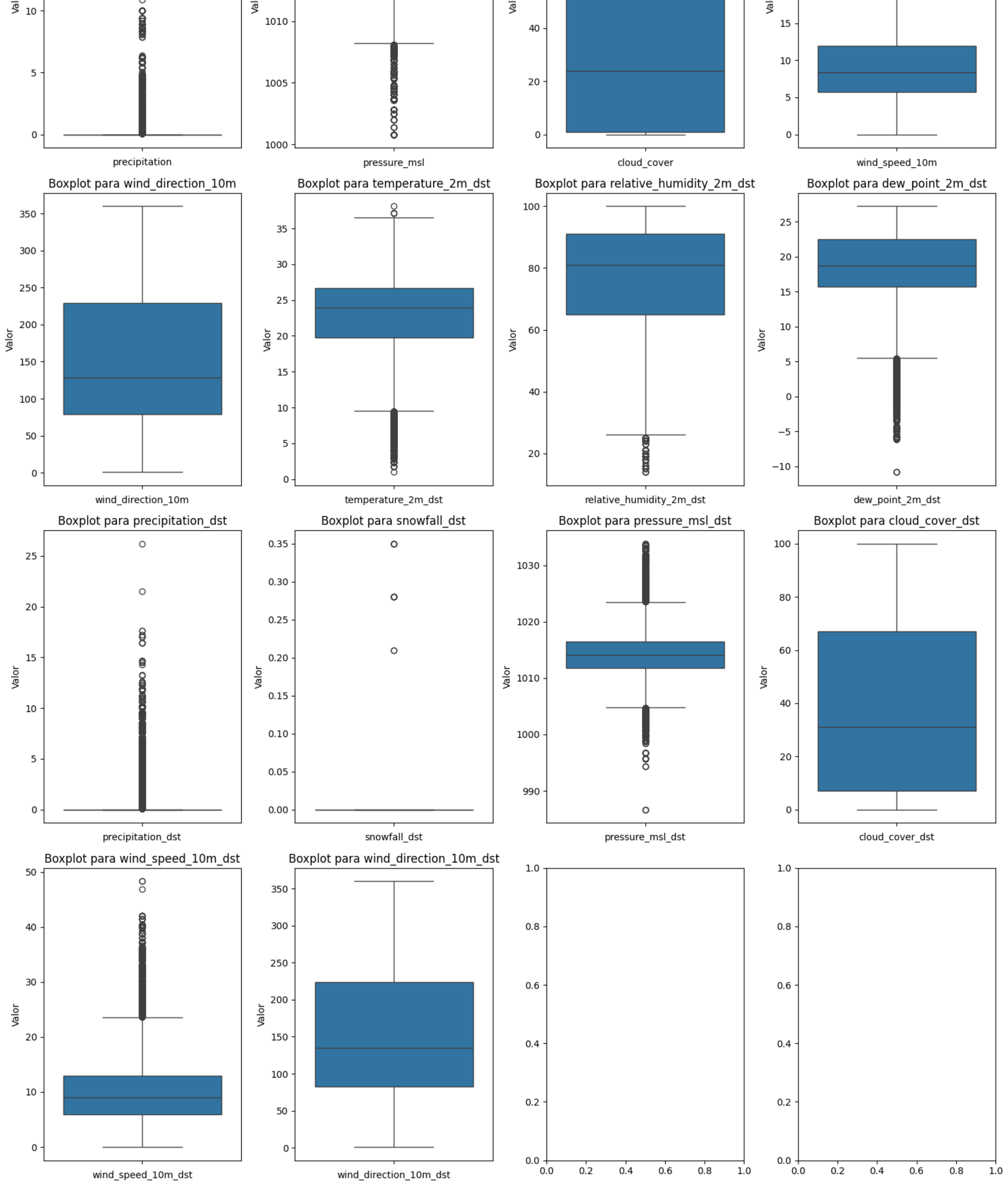
| | IATA code | Destination | Flight | Airline | Status | Origin | temperature |
|---|---|---|---|---|---|---|---|
| **count** | 102190.000000 | 102190.000000 | 102190.000000 | 102190.000000 | 102190.000000 | 102190.000000 | 102190.00 |
| **mean** | 74.581270 | 87.714855 | 2744.192308 | 73.160936 | 0.045220 | 3.863235 | 22.74 |
| **std** | 44.381904 | 47.754533 | 1486.453395 | 42.188062 | 0.207787 | 2.201285 | 4.49 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **min** | 0.000000 | 0.000000 | 0.000000 | -1.000000 | 0.000000 | 0.000000 | 12.30 |
| **25%** | 35.000000 | 41.000000 | 1508.000000 | 31.000000 | 0.000000 | 3.000000 | 19.10 |
| **50%** | 59.000000 | 99.000000 | 2824.000000 | 69.000000 | 0.000000 | 4.000000 | 22.70 |
| **75%** | 120.000000 | 129.000000 | 3995.750000 | 102.000000 | 0.000000 | 4.000000 | 26.40 |
| **max** | 161.000000 | 156.000000 | 5292.000000 | 153.000000 | 1.000000 | 9.000000 | 36.50 |

Uma das técnicas mais comuns para identificar outliers é o uso de boxplots. Nessa representação gráfica, a linha dentro da caixa representa a mediana dos dados, enquanto a caixa em si representa o intervalo interquartil, que abrange os valores entre o primeiro e o terceiro quartil. Os whiskers (ou "bigodes") estendem-se a partir da caixa e representam os limites do intervalo dos dados. Tipicamente, esses limites são calculados como o valor do intervalo interquartil multiplicado por 1.5. Qualquer ponto fora desse intervalo é considerado um outlier.

```
In [ ]:
fig, axs = plt.subplots(6, 4, figsize=(15, 5*6))
for i, column in enumerate(dfComplete.loc[:, ~dfComplete.columns.isin(['Status'])].colum
  ax = axs[i // 4, i % 4]
  sns.boxplot(data=dfComplete[column], ax=ax)
  ax.set_title(f'Boxplot para {column}')
  ax.set_xlabel(column)
  ax.set_ylabel('Valor')
plt.tight_layout()
plt.show()
```





Os outliers serão substituídos por NaN, de modo que sejam tratados como valores ausentes. Esta abordagem baseia-se na premissa de que outliers não contribuem positivamente para o treinamento do modelo, sendo equivalentes à ausência de dados.

Ao substituir outliers por NaN, garantimos que esses valores extremos não distorçam as análises estatísticas e a modelagem preditiva. Além disso, essa prática facilita o tratamento uniforme de dados problemáticos, permitindo a aplicação consistente de técnicas de imputação ou exclusão de valores ausentes.

## Z-Score Robusto

In [ ]:
```
outliers_values = 0
for column in dfComplete.loc[:, ~dfComplete.columns.isin(['Status'])].columns:
  data = dfComplete[column]
  mediana = np.median(data)
  mad = np.median(np.abs(data - mediana))
  zscore_robusto = 0.6745 * (data - mediana) / mad
  outliers_values_column = np.sum(np.abs(zscore_robusto) > 3.5)
  outliers_values += outliers_values_column
  print(f'{column}: N° de outliers {outliers_values_column} e razão {outliers_values_col
print(f'\nValor total de outliers: {outliers_values}')
```

```
IATA code: N° de outliers 0 e razão 0.0
Destination: N° de outliers 0 e razão 0.0
Flight: N° de outliers 0 e razão 0.0
Airline: N° de outliers 0 e razão 0.0
Origin: N° de outliers 0 e razão 0.0
temperature_2m: N° de outliers 0 e razão 0.0
relative_humidity_2m: N° de outliers 0 e razão 0.0
dew_point_2m: N° de outliers 45 e razão 0.0004403561992367159
precipitation: N° de outliers 14283 e razão 0.13976905763773365
pressure_msl: N° de outliers 200 e razão 0.001957138663274293
cloud_cover: N° de outliers 0 e razão 0.0
wind_speed_10m: N° de outliers 270 e razão 0.0026421371954202955
wind_direction_10m: N° de outliers 0 e razão 0.0
temperature_2m_dst: N° de outliers 375 e razão 0.0036696349936392995
relative_humidity_2m_dst: N° de outliers 24 e razão 0.00023485663959291515
dew_point_2m_dst: N° de outliers 435 e razão 0.0042567765926215875
precipitation_dst: N° de outliers 22473 e razão 0.21991388589881594
snowfall_dst: N° de outliers 11 e razão 0.00010764262648008612
pressure_msl_dst: N° de outliers 572 e razão 0.005597416576964478
cloud_cover_dst: N° de outliers 0 e razão 0.0
wind_speed_10m_dst: N° de outliers 848 e razão 0.008298267932283002
wind_direction_10m_dst: N° de outliers 0 e razão 0.0

Valor total de outliers: 39536
```

## Tukey

In [ ]:
```
def count_outliers(df, column):
  Q1 = df[column].quantile(0.25)
  Q3 = df[column].quantile(0.75)
  IQR = Q3 - Q1
  lower_bound = Q1 - 1.5 * IQR
  upper_bound = Q3 + 1.5 * IQR
  return np.sum((df[column] < lower_bound) | (df[column] > upper_bound))

outliers_values = 0
for column in dfComplete.loc[:, ~dfComplete.columns.isin(['Status'])].columns:
  outliers_values_column = count_outliers(dfComplete, column)
  outliers_values += outliers_values_column
  print(f'{column}: N° de outliers {outliers_values_column} e razão {outliers_values_col

print(f'\nValor total de outliers: {outliers_values}')
```

```
IATA code: N° de outliers 0 e razão 0.0
Destination: N° de outliers 0 e razão 0.0
Flight: N° de outliers 0 e razão 0.0
Airline: N° de outliers 0 e razão 0.0
Origin: N° de outliers 34592 e razão 0.3385067031999217
temperature_2m: N° de outliers 0 e razão 0.0
relative_humidity_2m: N° de outliers 44 e razão 0.00043057050592O3445
dew_point_2m: N° de outliers 117 e razão 0.0011449261180154614
precipitation: N° de outliers 14283 e razão 0.13976905763773365
pressure_msl: N° de outliers 486 e razão 0.004755846951756532
```

```
cloud_cover: N° de outliers 0 e razão 0.0
wind_speed_10m: N° de outliers 1221 e razão 0.011948331539289559
wind_direction_10m: N° de outliers 0 e razão 0.0
temperature_2m_dst: N° de outliers 1063 e razão 0.010402191995302868
relative_humidity_2m_dst: N° de outliers 183 e razão 0.0017907818768959781
dew_point_2m_dst: N° de outliers 2002 e razão 0.019590958019375673
precipitation_dst: N° de outliers 22473 e razão 0.21991388589881594
snowfall_dst: N° de outliers 11 e razão 0.00010764262648008612
pressure_msl_dst: N° de outliers 1323 e razão 0.012946472257559448
cloud_cover_dst: N° de outliers 0 e razão 0.0
wind_speed_10m_dst: N° de outliers 1889 e razão 0.018485174674625696
wind_direction_10m_dst: N° de outliers 0 e razão 0.0

Valor total de outliers: 79687
```

Apesar do Z-Score Robusto ser mais adequado para dados que não seguem uma distribuição normal, ele é mais sensível a valores extremos e requer definição de um limite para identificar outliers, o que pode ser subjetivo. Já o Método de Tukey (IQR) Assume uma distribuição normal dos dados, é uma abordagem mais simples e amplamente utilizada.

In [ ]:
```python
def remove_outliers(df, column):
  Q1 = df[column].quantile(0.25)
  Q3 = df[column].quantile(0.75)
  IQR = Q3 - Q1
  lower_bound = Q1 - 1.5 * IQR
  upper_bound = Q3 + 1.5 * IQR
  return np.where((df[column] < lower_bound) | (df[column] > upper_bound), np.nan, df[co

df_clean = pd.DataFrame()
for column in dfComplete.loc[:, ~dfComplete.columns.isin(['Status', 'IATA code', 'Origin
  df_clean[column] = remove_outliers(dfComplete, column)
```

In [ ]:
```python
df_clean['Status'] = dfComplete['Status']
df_clean['IATA code'] = dfComplete['IATA code']
df_clean['Origin'] = dfComplete['Origin']
```

## Isolation Florest

In [ ]:
```python
clf = IsolationForest(max_samples=100, random_state=42)
clf.fit(dfComplete.loc[:, ~dfComplete.columns.isin(['Status'])])
```

Out[ ]:
```
IsolationForest
IsolationForest(max_samples=100, random_state=42)
```

In [ ]:
```python
scores = clf.predict(dfComplete.loc[:, ~dfComplete.columns.isin(['Status'])])
```

In [ ]:
```python
dfComplete_clf = dfComplete.copy()
dfComplete_clf['outlier'] = scores
dfComplete_clf[dfComplete_clf['outlier'] == -1]
```

Out[ ]:

| | IATA code | Destination | Flight | Airline | Status | Origin | temperature_2m | relative_humidity_2m | dew_point_2m |
|---|---|---|---|---|---|---|---|---|---|
| **2** | 40 | 41 | 4391 | 117 | 0 | 4 | 22.6 | 72 | 17.3 |
| **3** | 40 | 41 | 2785 | 68 | 0 | 1 | 22.5 | 76 | 18.0 |
| **4** | 40 | 41 | 3717 | 99 | 0 | 1 | 22.5 | 76 | 18.0 |
| **17** | 134 | 136 | 4181 | 102 | 0 | 4 | 22.6 | 72 | 17.3 |
| **18** | 134 | 136 | 1790 | 42 | 0 | 4 | 22.6 | 72 | 17.3 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **126294** | 86 | 74 | 4600 | 138 | 0 | 8 | 27.0 | 70 | 20.9 |
| **126295** | 86 | 74 | 991 | 30 | 0 | 8 | 27.0 | 70 | 20.9 |
| **126296** | 86 | 74 | 2591 | 58 | 0 | 8 | 27.0 | 70 | 20.9 |
| **126298** | 154 | 25 | 515 | 31 | 0 | 8 | 26.7 | 74 | 21.6 |
| **126299** | 154 | 25 | 4953 | 136 | 0 | 8 | 26.7 | 74 | 21.6 |

24853 rows × 24 columns

De acordo com a análise global do Isolation Florest, a qual considera múltiplas variáveis, quase mais vinte mil linhas devem ser descartadas por não representarem a distribuição dos dados, serem outliers. No entanto, a remoção não será realizada, dada a preocupação com os desbalanceamento dos dados. Buscamos evitar remoção de qualquer linha das classes minoritárias para ter o máximo de informações disponíveis.

## Tratamento de Dados Ausentes

A identificação e tratamento dos valores ausentes no dataset são etapas essenciais no processo de pré-processamento de dados. É crucial determinar a porcentagem de dados faltantes em cada coluna. Essa análise permite avaliar a extensão do problema e decidir sobre a melhor abordagem para lidar com os dados ausentes.

Até um determinado limiar de porcentagem de dados faltantes, é viável aplicar estratégias de imputação ou preenchimento dos valores ausentes. Essas estratégias podem incluir a substituição dos valores ausentes por estatísticas descritivas, como a média, mediana ou moda da coluna, ou por valores previamente estabelecidos com base no conhecimento do domínio.

No entanto, acima de um certo limiar de porcentagem de dados faltantes, a coluna em questão pode perder seu poder informativo e relevância para a análise. Nesses casos, a remoção da coluna é frequentemente a abordagem mais apropriada, uma vez que manter colunas com uma quantidade significativa de valores ausentes pode distorcer os resultados da análise e prejudicar a eficácia do modelo.

In [ ]:
```
dfComplete_clf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 102190 entries, 0 to 126302
Data columns (total 24 columns):
 #   Column                   Non-Null Count   Dtype  
---  ------                   --------------   -----  
 0   IATA code                102190 non-null  int16  
 1   Destination              102190 non-null  int16  
 2   Flight                   102190 non-null  int16  
 3   Airline                  102190 non-null  int16  
 4   Status                   102190 non-null  int64  
 5   Origin                   102190 non-null  int8   
 6   temperature_2m           102190 non-null  float64
 7   relative_humidity_2m     102190 non-null  int64  
 8   dew_point_2m             102190 non-null  float64
 9   precipitation            102190 non-null  float64
 10  pressure_msl             102190 non-null  float64
 11  cloud_cover              102190 non-null  int64  
 12  wind_speed_10m           102190 non-null  float64
 13  wind_direction_10m        102190 non-null  int64
 14  temperature_2m_dst        102190 non-null  float64
 15  relative_humidity_2m_dst  102190 non-null  int64
 16  dew_point_2m_dst          102190 non-null  float64
 17  precipitation_dst         102190 non-null  float64
 18  snowfall_dst              102190 non-null  float64
 19  pressure_msl_dst          102190 non-null  float64
 20  cloud_cover_dst           102190 non-null  int64
 21  wind_speed_10m_dst        102190 non-null  float64
 22  wind_direction_10m_dst    102190 non-null  int64
 23  outlier                   102190 non-null  int64
dtypes: float64(11), int16(4), int64(8), int8(1)
memory usage: 20.5 MB
```

## Colunas

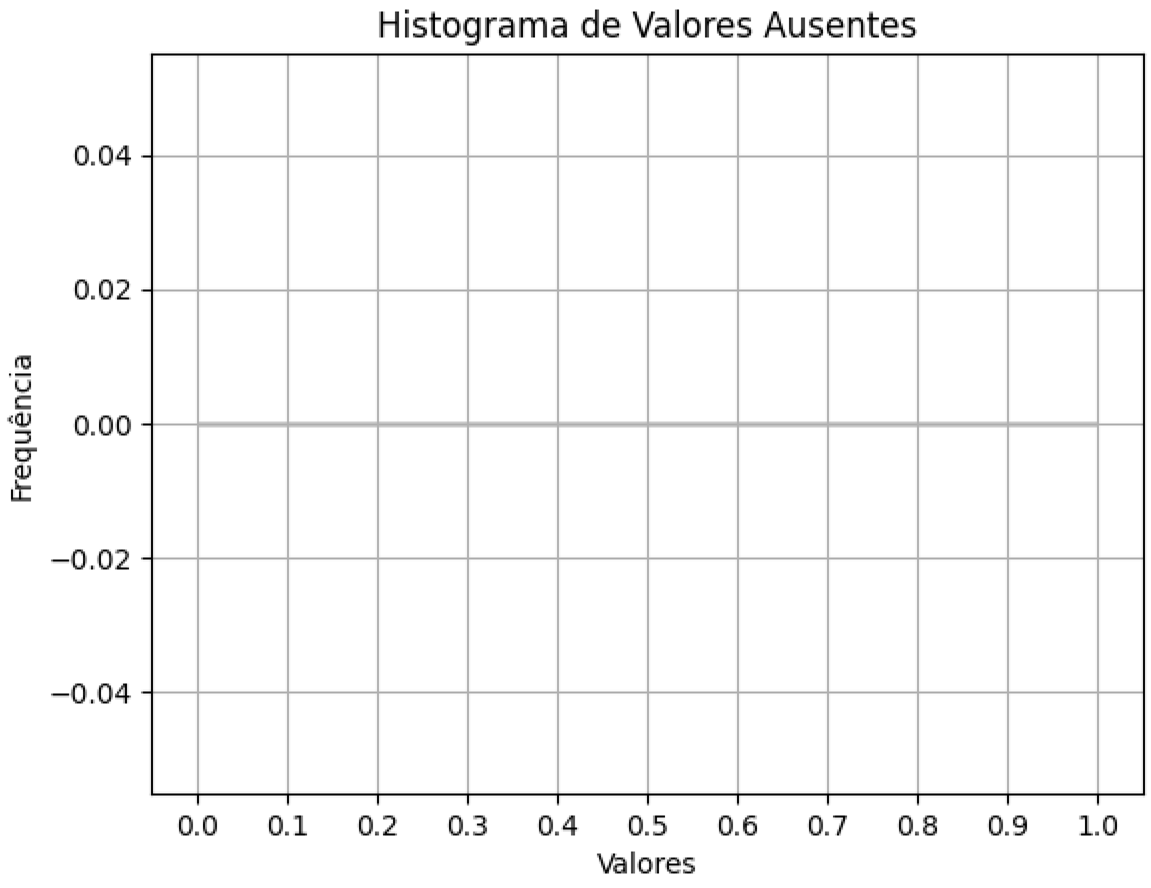
In [ ]:
```
null_values = dfComplete_clf.isnull().sum() / len(dfComplete_clf)
null_values = null_values[null_values != 0]
null_values
```

Out[ ]:
```
Series([], dtype: float64)
```

In [ ]:
```
null_values = dfComplete_clf.isnull().sum() / len(dfComplete_clf)
null_values = null_values[null_values != 0]
null_values
```

Out[ ]:
```
Series([], dtype: float64)
```

In [ ]:
```
plt.hist(null_values, bins=10, range=(0, 1), edgecolor='black')
plt.title('Histograma de Valores Ausentes')
plt.xlabel('Valores')
plt.ylabel('Frequência')
plt.xticks(np.arange(0, 1.1, 0.1))
plt.grid(True)
plt.show()
```



Após a análise do histograma e a definição do limiar de 30% para valores ausentes, optamos por remover todas as colunas em que mais há mais de 30% de dados faltantes. Essa decisão visa simplificar o treinamento do modelo e evitar a distorção da distribuição natural dos dados devido à necessidade de imputação de uma grande quantidade de valores ausentes.

Essa abordagem de pré-processamento permite reduzir a complexidade do modelo e preservar a integridade dos dados, concentrando-se em variáveis mais informativas e relevantes para a análise. Dessa forma, garantimos uma representação mais precisa e eficiente dos dados, contribuindo para a qualidade e a robustez do modelo resultante.

In [ ]:
```
df_clean2 = dfComplete_clf.dropna(thresh=0.3*len(dfComplete_clf), axis=1)
dfComplete_clf.shape, df_clean2.shape
```

Out[ ]:
```
((102190, 24), (102190, 24))
```

In [ ]:
```
colunas_valores_iguais = dfComplete_clf.columns[dfComplete_clf.apply(lambda x: x.nunique
print(colunas_valores_iguais)

dfComplete_clf = dfComplete_clf.drop(columns=colunas_valores_iguais)
```

```
[]
```

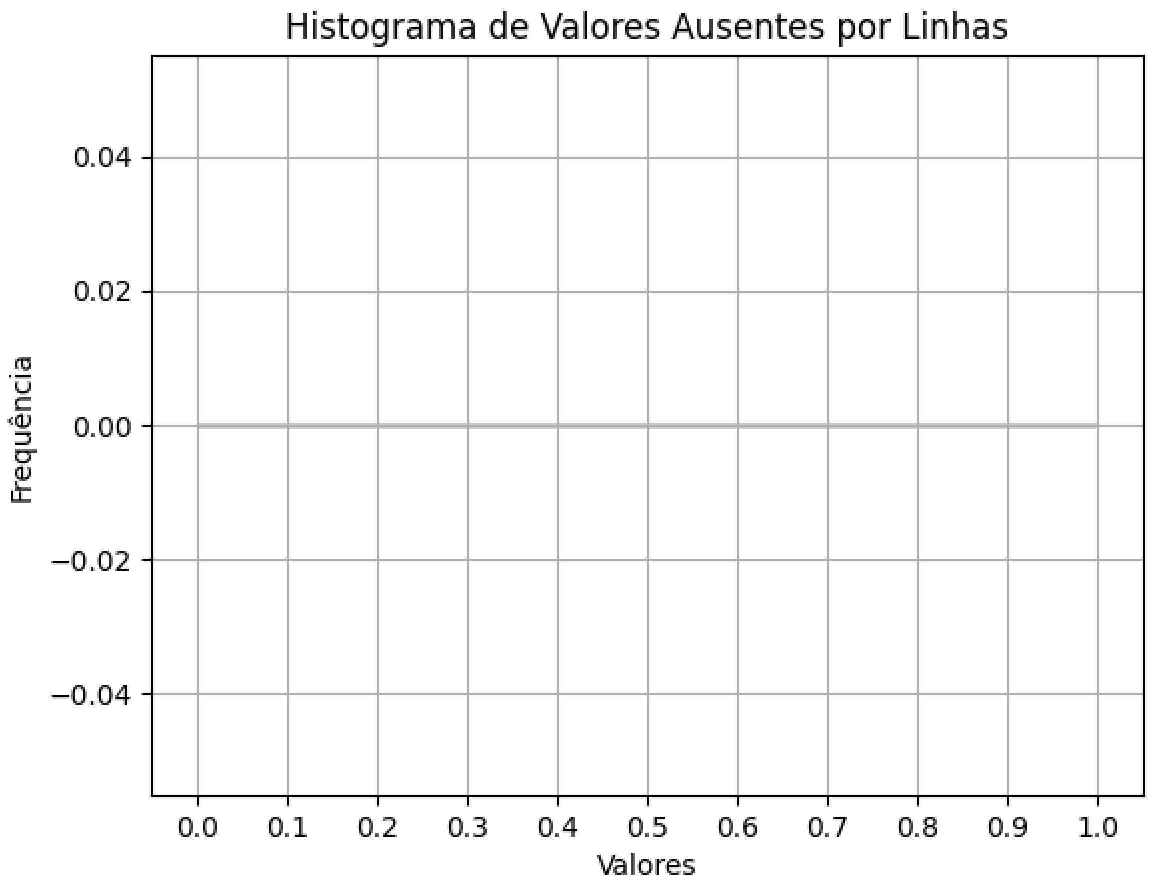
## Linhas

Uma segunda análise é conduzida para verificar a quantidade de dados ausentes por instância. Caso essa quantidade supere o limiar estabelecido, considera-se que aquela amostra contribui pouco para a compreensão do problema real em questão.

In [ ]:
```
null_values_lines = dfComplete_clf.isnull().sum(axis=1) / len(dfComplete_clf.columns)
null_values_lines = null_values_lines[null_values_lines != 0]
null_values_lines
```

Out[ ]:
```
Series([], dtype: float64)
```

In [ ]:
```
plt.hist(null_values_lines, bins=10, range=(0, 1), edgecolor='black')
plt.title('Histograma de Valores Ausentes por Linhas')
plt.xlabel('Valores')
plt.ylabel('Frequência')
plt.xticks(np.arange(0, 1.1, 0.1))
plt.grid(True)
plt.show()
```



In [ ]:
```
df_clean2 = dfComplete_clf.dropna(thresh=0.3*len(dfComplete_clf.columns), axis=0)
dfComplete_clf.shape, df_clean2.shape
```

Out[ ]:
```
((102190, 24), (102190, 24))
```

## Remoção dos Valores Ausentes

O KNNImputer é uma classe da biblioteca scikit-learn usada para imputar valores ausentes em conjuntos de dados. Ele preenche os valores ausentes considerando os valores dos K vizinhos mais próximos que são mais semelhantes em termos das características observadas. Cada valor ausente é substituído pela média dos valores correspondentes dos vizinhos mais próximos.

As principais vantagens do KNNImputer incluem sua capacidade de aproveitar as relações e similaridades entre as amostras, resultando em estimativas mais precisas em comparação com técnicas simples, como substituição por média ou mediana. No entanto, o KNNImputer pode ser computacionalmente intensivo para conjuntos de dados muito grandes, o que pode ser uma limitação em termos de tempo e recursos computacionais. Por exemplo, foram necessários treze minutos para a realização dessa etapa pelo Colab.

In [ ]:
```
df_imputed = pd.DataFrame(KNNImputer(n_neighbors=3).fit_transform(dfComplete_clf), colum
df_imputed
```

Out[ ]:

| | IATA code | Destination | Flight | Airline | Status | Origin | temperature_2m | relative_humidity_2m | dew_point_2r |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 40.0 | 41.0 | 3741.0 | 101.0 | 0.0 | 4.0 | 22.6 | 72.0 | 17. |
| **1** | 40.0 | 41.0 | 2425.0 | 53.0 | 0.0 | 4.0 | 22.6 | 72.0 | 17. |
| **2** | 40.0 | 41.0 | 4391.0 | 117.0 | 0.0 | 4.0 | 22.6 | 72.0 | 17. |
| **3** | 40.0 | 41.0 | 2785.0 | 68.0 | 0.0 | 1.0 | 22.5 | 76.0 | 18. |
| **4** | 40.0 | 41.0 | 3717.0 | 99.0 | 0.0 | 1.0 | 22.5 | 76.0 | 18. |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | . |
| **102185** | 154.0 | 25.0 | 515.0 | 31.0 | 0.0 | 8.0 | 26.7 | 74.0 | 21. |
| **102186** | 154.0 | 25.0 | 4953.0 | 136.0 | 0.0 | 8.0 | 26.7 | 74.0 | 21. |
| **102187** | 56.0 | 124.0 | 3904.0 | 101.0 | 0.0 | 8.0 | 26.8 | 71.0 | 21. |
| **102188** | 56.0 | 124.0 | 2374.0 | 53.0 | 0.0 | 8.0 | 26.8 | 71.0 | 21. |
| **102189** | 56.0 | 124.0 | 4244.0 | 110.0 | 0.0 | 8.0 | 26.8 | 71.0 | 21. |

102190 rows × 24 columns

In [ ]:
```
df_imputed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 102190 entries, 0 to 102189
Data columns (total 24 columns):
 #   Column                   Non-Null Count   Dtype  
---  ------                   --------------   -----  
 0   IATA code                102190 non-null  float64
 1   Destination              102190 non-null  float64
 2   Flight                   102190 non-null  float64
 3   Airline                  102190 non-null  float64
 4   Status                   102190 non-null  float64
 5   Origin                   102190 non-null  float64
 6   temperature_2m           102190 non-null  float64
 7   relative_humidity_2m     102190 non-null  float64
 8   dew_point_2m             102190 non-null  float64
 9   precipitation            102190 non-null  float64
 10  pressure_msl             102190 non-null  float64
 11  cloud_cover              102190 non-null  float64
 12  wind_speed_10m           102190 non-null  float64
 13  wind_direction_10m       102190 non-null  float64
 14  temperature_2m_dst       102190 non-null  float64
 15  relative_humidity_2m_dst 102190 non-null  float64
 16  dew_point_2m_dst         102190 non-null  float64
 17  precipitation_dst        102190 non-null  float64
 18  snowfall_dst             102190 non-null  float64
 19  pressure_msl_dst         102190 non-null  float64
 20  cloud_cover_dst          102190 non-null  float64
 21  wind_speed_10m_dst       102190 non-null  float64
 22  wind_direction_10m_dst   102190 non-null  float64
 23  outlier                  102190 non-null  float64
dtypes: float64(24)
memory usage: 18.7 MB
```

In [ ]:
```
df_imputed.to_csv('/content/df_imputed.csv', index=False)
```

## Normalização e Discretização

Para garantir que nenhuma variável tenha uma influência desproporcional no treinamento do modelo, é essencial normalizar todo o conjunto de dados.

Utilizaremos a técnica de Min-Max Scaling, que ajusta os valores para um intervalo entre 0 e 1, preservando suas distribuições relativas. A normalização pelo método Min-Max Scaling transforma os dados de modo que o valor mínimo de cada variável se torne 0 e o valor máximo se torne 1.

Essa abordagem assegura que todas as variáveis contribuam de maneira equilibrada durante o treinamento, evitando que variáveis com magnitudes maiores dominem o processo de aprendizado. Além disso, a normalização mantém a distribuição original dos dados, permitindo que o modelo capture as relações intrínsecas de maneira eficaz.

In [ ]:
```
df_imputed = pd.read_csv('/content/df_imputed.csv')
df_imputed
```

Out[ ]:

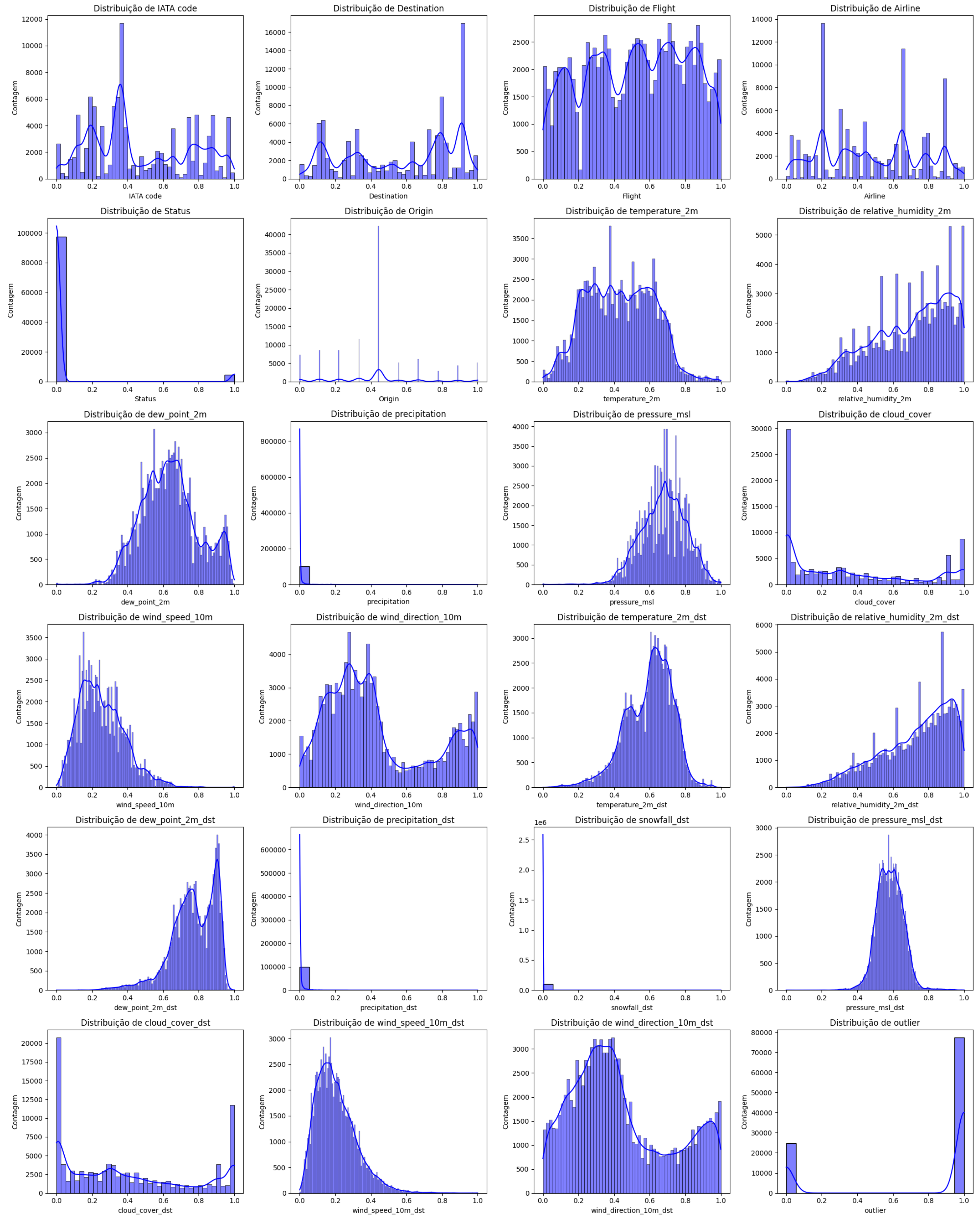
| | IATA code | Destination | Flight | Airline | Status | Origin | temperature_2m | relative_humidity_2m | dew_point_2r |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 40.0 | 41.0 | 3741.0 | 101.0 | 0.0 | 4.0 | 22.6 | 72.0 | 17. |
| **1** | 40.0 | 41.0 | 2425.0 | 53.0 | 0.0 | 4.0 | 22.6 | 72.0 | 17. |
| **2** | 40.0 | 41.0 | 4391.0 | 117.0 | 0.0 | 4.0 | 22.6 | 72.0 | 17. |
| **3** | 40.0 | 41.0 | 2785.0 | 68.0 | 0.0 | 1.0 | 22.5 | 76.0 | 18. |
| **4** | 40.0 | 41.0 | 3717.0 | 99.0 | 0.0 | 1.0 | 22.5 | 76.0 | 18. |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | . |
| **102185** | 154.0 | 25.0 | 515.0 | 31.0 | 0.0 | 8.0 | 26.7 | 74.0 | 21. |
| **102186** | 154.0 | 25.0 | 4953.0 | 136.0 | 0.0 | 8.0 | 26.7 | 74.0 | 21. |
| **102187** | 56.0 | 124.0 | 3904.0 | 101.0 | 0.0 | 8.0 | 26.8 | 71.0 | 21. |
| **102188** | 56.0 | 124.0 | 2374.0 | 53.0 | 0.0 | 8.0 | 26.8 | 71.0 | 21. |
| **102189** | 56.0 | 124.0 | 4244.0 | 110.0 | 0.0 | 8.0 | 26.8 | 71.0 | 21. |

102190 rows × 24 columns

In [ ]:
```
scaler = MinMaxScaler()
df_normalized = scaler.fit_transform(df_imputed)
df_normalized = pd.DataFrame(df_normalized, columns=df_imputed.columns)
```

**Exemplo de como a Discretização pode ser utilizada**

In [ ]:
```
fig, axs = plt.subplots(6, 4, figsize=(20, 5 * 5))
for i, coluna in enumerate(df_normalized.columns):
  ax = axs[i // 4, i % 4]

  sns.histplot(data=df_normalized, x=coluna, ax=ax, color='blue', alpha=0.5, kde=True)

  ax.set_title(f'Distribuição de {coluna}')
  ax.set_xlabel(f'{coluna}')
  ax.set_ylabel('Contagem')
plt.tight_layout()
plt.show()
```



```
In [ ]: pd.cut(df_normalized['cloud_cover'], 10).value_counts().describe()
```

```
Out[ ]: count       10.000000
        mean     10219.000000
        std      10540.484313
        min       2681.000000
        25%       5021.500000
        50%       7431.500000
        75%       9497.250000
```

```
max      38899.000000
Name: count, dtype: float64
```

In [ ]:
```
pd.qcut(df_normalized['cloud_cover'], 4).value_counts().describe()
```

Out[ ]:
```
count        4.000000
mean     25547.500000
std       1301.977598
min      24214.000000
25%      24925.750000
50%      25326.000000
75%      25947.750000
max      27324.000000
Name: count, dtype: float64
```

In [ ]:
```
pd.cut(df_normalized['cloud_cover_dst'], 10).value_counts().describe()
```

Out[ ]:
```
count       10.00000
mean     10219.00000
std       7548.12402
min       3191.00000
25%       5627.50000
50%       8621.50000
75%      10927.50000
max      29174.00000
Name: count, dtype: float64
```

In [ ]:
```
pd.qcut(df_normalized['cloud_cover_dst'], 4).value_counts().describe()
```

Out[ ]:
```
count        4.000000
mean     25547.500000
std        466.096199
min      25034.000000
25%      25379.750000
50%      25495.000000
75%      25662.750000
max      26166.000000
Name: count, dtype: float64
```

In [ ]:
```
df_normalized.to_csv('/content/df_normalized.csv', index=False)
```

## Testes de Hipóteses

- Comparação de valores de categorias e visualizar diferenças

In [ ]:
```
dfComplete = pd.read_csv('/content/df_normalized.csv')
dfComplete
```

Out[ ]:

| | IATA code | Destination | Flight | Airline | Status | Origin | temperature_2m | relative_humidity_2m | de |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.248447 | 0.262821 | 0.706916 | 0.662338 | 0.0 | 0.444444 | 0.425620 | 0.662651 | |
| **1** | 0.248447 | 0.262821 | 0.458239 | 0.350649 | 0.0 | 0.444444 | 0.425620 | 0.662651 | |
| **2** | 0.248447 | 0.262821 | 0.829743 | 0.766234 | 0.0 | 0.444444 | 0.425620 | 0.662651 | |
| **3** | 0.248447 | 0.262821 | 0.526266 | 0.448052 | 0.0 | 0.111111 | 0.421488 | 0.710843 | |
| **4** | 0.248447 | 0.262821 | 0.702381 | 0.649351 | 0.0 | 0.111111 | 0.421488 | 0.710843 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **102185** | 0.956522 | 0.160256 | 0.097317 | 0.207792 | 0.0 | 0.888889 | 0.595041 | 0.686747 | |
| **102186** | 0.956522 | 0.160256 | 0.935941 | 0.889610 | 0.0 | 0.888889 | 0.595041 | 0.686747 | |

|  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|
| **102187** | 0.347826 | 0.794872 | 0.737717 | 0.662338 | 0.0 | 0.888889 | 0.599174 | 0.650602 |
| **102188** | 0.347826 | 0.794872 | 0.448602 | 0.350649 | 0.0 | 0.888889 | 0.599174 | 0.650602 |
| **102189** | 0.347826 | 0.794872 | 0.801965 | 0.720779 | 0.0 | 0.888889 | 0.599174 | 0.650602 |

102190 rows × 24 columns

O teste de hipótese é um procedimento estatístico usado para tomar decisões sobre a validade de uma afirmação (hipótese) com base em dados amostrais. Envolve formular duas hipóteses: a hipótese nula (H0), que representa a situação atual ou uma posição de não-efeito, e a hipótese alternativa (H1), que representa uma mudança ou um efeito. O teste então calcula a probabilidade (valor p) de observar os dados amostrais se a hipótese nula fosse verdadeira. Com base no valor p e em um nível de significância predefinido (geralmente 0,05), decidimos se rejeitamos ou não a hipótese nula, ajudando a inferir se há evidência estatística suficiente para apoiar a hipótese alternativa.

- Para cada variável, dividimos os dados em duas amostras: voos cancelados e voos não cancelados.
- Teste de Normalidade (teste de Shapiro-Wilk).
- Escolha do Teste Estatístico: Dependendo do resultado do teste de normalidade, escolhemos entre o teste t de Student (para distribuições normais) e o teste de Mann-Whitney U (para distribuições não normais).
- Comparar o valor p para decidir se rejeitamos ou não a hipótese nula.

In [ ]:
```
dfComplete.columns
```

Out[ ]:
```
Index(['IATA code', 'Destination', 'Flight', 'Airline', 'Status', 'Origin',
       'temperature_2m', 'relative_humidity_2m', 'dew_point_2m',
       'precipitation', 'pressure_msl', 'cloud_cover', 'wind_speed_10m',
       'wind_direction_10m', 'temperature_2m_dst', 'relative_humidity_2m_dst',
       'dew_point_2m_dst', 'precipitation_dst', 'snowfall_dst',
       'pressure_msl_dst', 'cloud_cover_dst', 'wind_speed_10m_dst',
       'wind_direction_10m_dst', 'outlier'],
      dtype='object')
```

In [ ]:
```
alpha = 0.05

colunas_clima = [
    'temperature_2m', 'relative_humidity_2m', 'dew_point_2m', 'pressure_msl', 'cloud_cov
    'wind_speed_10m', 'wind_direction_10m', 'temperature_2m_dst', 'relative_humidity_2m_
    'dew_point_2m_dst', 'pressure_msl_dst', 'cloud_cover_dst', 'wind_speed_10m_dst', 'wi
]

def realizar_teste(coluna):
    cancelados = dfComplete[dfComplete["Status"] == 1][coluna]
    nao_cancelados = dfComplete[dfComplete["Status"] != 0][coluna]

    shapiro_cancelados = shapiro(cancelados)
    shapiro_nao_cancelados = shapiro(nao_cancelados)

    if shapiro_cancelados.pvalue > alpha and shapiro_nao_cancelados.pvalue > alpha:
        t_stat, p_value = ttest_ind(cancelados, nao_cancelados)
        teste = "t de Student"
    else:
        u_stat, p_value = mannwhitneyu(cancelados, nao_cancelados)
        teste = "Mann-Whitney U"

    if p_value < alpha:
        resultado = f"Rejeitar a hipótese nula: Há diferença significativa na taxa de vo
    else:
        resultado = f"Não rejeitar a hipótese nula: Não há diferença significativa na ta

    return coluna, teste, p_value, resultado

resultados = [realizar_teste(coluna) for coluna in colunas_clima]

# Exibir os resultados
for coluna, teste, p_value, resultado in resultados:
    print(f"Variável: {coluna}")
    print(f"Hipótese Nula (H0): Não há diferença na taxa de voos atrasados ou cancelados
    print(f"Hipótese Alternativa (H1): Há diferença na taxa de voos atrasados ou cancela
    print(f"Teste escolhido: {teste}")
    print(f"p-value: {p_value}")
    print(resultado)
    print("-" * 80)
```

```
Variável: temperature_2m
Hipótese Nula (H0): Não há diferença na taxa de voos atrasados ou cancelados com relação
à variável climática temperature_2m.
Hipótese Alternativa (H1): Há diferença na taxa de voos atrasados ou cancelados com rela
ção à variável climática temperature_2m.
Teste escolhido: Mann-Whitney U
p-value: 1.0
Não rejeitar a hipótese nula: Não há diferença significativa na taxa de voos atrasados o
u cancelados com relação a temperature_2m.
--------------------------------------------------------------------------------
Variável: relative_humidity_2m
Hipótese Nula (H0): Não há diferença na taxa de voos atrasados ou cancelados com relação
à variável climática relative_humidity_2m.
Hipótese Alternativa (H1): Há diferença na taxa de voos atrasados ou cancelados com rela
ção à variável climática relative_humidity_2m.
Teste escolhido: Mann-Whitney U
p-value: 1.0
Não rejeitar a hipótese nula: Não há diferença significativa na taxa de voos atrasados o
u cancelados com relação a relative_humidity_2m.
--------------------------------------------------------------------------------
Variável: dew_point_2m
Hipótese Nula (H0): Não há diferença na taxa de voos atrasados ou cancelados com relação
à variável climática dew_point_2m.
Hipótese Alternativa (H1): Há diferença na taxa de voos atrasados ou cancelados com rela
ção à variável climática dew_point_2m.
Teste escolhido: Mann-Whitney U
p-value: 1.0
Não rejeitar a hipótese nula: Não há diferença significativa na taxa de voos atrasados o
u cancelados com relação a dew_point_2m.
--------------------------------------------------------------------------------
Variável: pressure_msl
Hipótese Nula (H0): Não há diferença na taxa de voos atrasados ou cancelados com relação
à variável climática pressure_msl.
Hipótese Alternativa (H1): Há diferença na taxa de voos atrasados ou cancelados com rela
ção à variável climática pressure_msl.
Teste escolhido: Mann-Whitney U
p-value: 1.0
Não rejeitar a hipótese nula: Não há diferença significativa na taxa de voos atrasados o
u cancelados com relação a pressure_msl.
--------------------------------------------------------------------------------
Variável: cloud_cover
Hipótese Nula (H0): Não há diferença na taxa de voos atrasados ou cancelados com relação
à variável climática cloud_cover.
Hipótese Alternativa (H1): Há diferença na taxa de voos atrasados ou cancelados com rela
ção à variável climática cloud_cover.
Teste escolhido: Mann-Whitney U
p-value: 1.0
Não rejeitar a hipótese nula: Não há diferença significativa na taxa de voos atrasados o
u cancelados com relação a cloud_cover.
--------------------------------------------------------------------------------
```

```
Variável: wind_speed_10m
Hipótese Nula (H0): Não há diferença na taxa de voos atrasados ou cancelados com relação
à variável climática wind_speed_10m.
Hipótese Alternativa (H1): Há diferença na taxa de voos atrasados ou cancelados com rela
ção à variável climática wind_speed_10m.
Teste escolhido: Mann-Whitney U
p-value: 1.0
Não rejeitar a hipótese nula: Não há diferença significativa na taxa de voos atrasados o
u cancelados com relação a wind_speed_10m.
--------------------------------------------------------------------------------
Variável: wind_direction_10m
Hipótese Nula (H0): Não há diferença na taxa de voos atrasados ou cancelados com relação
à variável climática wind_direction_10m.
Hipótese Alternativa (H1): Há diferença na taxa de voos atrasados ou cancelados com rela
ção à variável climática wind_direction_10m.
Teste escolhido: Mann-Whitney U
p-value: 1.0
Não rejeitar a hipótese nula: Não há diferença significativa na taxa de voos atrasados o
u cancelados com relação a wind_direction_10m.
--------------------------------------------------------------------------------
Variável: temperature_2m_dst
Hipótese Nula (H0): Não há diferença na taxa de voos atrasados ou cancelados com relação
à variável climática temperature_2m_dst.
Hipótese Alternativa (H1): Há diferença na taxa de voos atrasados ou cancelados com rela
ção à variável climática temperature_2m_dst.
Teste escolhido: Mann-Whitney U
p-value: 1.0
Não rejeitar a hipótese nula: Não há diferença significativa na taxa de voos atrasados o
u cancelados com relação a temperature_2m_dst.
--------------------------------------------------------------------------------
Variável: relative_humidity_2m_dst
Hipótese Nula (H0): Não há diferença na taxa de voos atrasados ou cancelados com relação
à variável climática relative_humidity_2m_dst.
Hipótese Alternativa (H1): Há diferença na taxa de voos atrasados ou cancelados com rela
ção à variável climática relative_humidity_2m_dst.
Teste escolhido: Mann-Whitney U
p-value: 1.0
Não rejeitar a hipótese nula: Não há diferença significativa na taxa de voos atrasados o
u cancelados com relação a relative_humidity_2m_dst.
--------------------------------------------------------------------------------
Variável: dew_point_2m_dst
Hipótese Nula (H0): Não há diferença na taxa de voos atrasados ou cancelados com relação
à variável climática dew_point_2m_dst.
Hipótese Alternativa (H1): Há diferença na taxa de voos atrasados ou cancelados com rela
ção à variável climática dew_point_2m_dst.
Teste escolhido: Mann-Whitney U
p-value: 1.0
Não rejeitar a hipótese nula: Não há diferença significativa na taxa de voos atrasados o
u cancelados com relação a dew_point_2m_dst.
--------------------------------------------------------------------------------
Variável: pressure_msl_dst
Hipótese Nula (H0): Não há diferença na taxa de voos atrasados ou cancelados com relação
à variável climática pressure_msl_dst.
Hipótese Alternativa (H1): Há diferença na taxa de voos atrasados ou cancelados com rela
ção à variável climática pressure_msl_dst.
Teste escolhido: Mann-Whitney U
p-value: 1.0
Não rejeitar a hipótese nula: Não há diferença significativa na taxa de voos atrasados o
u cancelados com relação a pressure_msl_dst.
--------------------------------------------------------------------------------
Variável: cloud_cover_dst
Hipótese Nula (H0): Não há diferença na taxa de voos atrasados ou cancelados com relação
à variável climática cloud_cover_dst.
Hipótese Alternativa (H1): Há diferença na taxa de voos atrasados ou cancelados com rela
ção à variável climática cloud_cover_dst.
Teste escolhido: Mann-Whitney U
```

```
p-value: 1.0
Não rejeitar a hipótese nula: Não há diferença significativa na taxa de voos atrasados o
u cancelados com relação a cloud_cover_dst.
--------------------------------------------------------------------------------
Variável: wind_speed_10m_dst
Hipótese Nula (H0): Não há diferença na taxa de voos atrasados ou cancelados com relação
à variável climática wind_speed_10m_dst.
Hipótese Alternativa (H1): Há diferença na taxa de voos atrasados ou cancelados com rela
ção à variável climática wind_speed_10m_dst.
Teste escolhido: Mann-Whitney U
p-value: 1.0
Não rejeitar a hipótese nula: Não há diferença significativa na taxa de voos atrasados o
u cancelados com relação a wind_speed_10m_dst.
--------------------------------------------------------------------------------
Variável: wind_direction_10m_dst
Hipótese Nula (H0): Não há diferença na taxa de voos atrasados ou cancelados com relação
à variável climática wind_direction_10m_dst.
Hipótese Alternativa (H1): Há diferença na taxa de voos atrasados ou cancelados com rela
ção à variável climática wind_direction_10m_dst.
Teste escolhido: Mann-Whitney U
p-value: 1.0
Não rejeitar a hipótese nula: Não há diferença significativa na taxa de voos atrasados o
u cancelados com relação a wind_direction_10m_dst.
--------------------------------------------------------------------------------
```

O número de voos cancelados é muito menor do que o número de voos não cancelados, isso pode afetar os testes de normalidade. Pequenas amostras são menos prováveis de passar nos testes de normalidade, o que pode resultar em uma escolha mais frequente do teste Mann-Whitney U.

O teste Mann-Whitney U é mais robusto em relação a outliers e distribuições não normais, tornando-o uma escolha segura quando há dúvidas sobre a normalidade dos dados ou quando os dados são muito desbalanceados.

## Parte 2: Classificação

In [5]:
```python
dfComplete = pd.read_csv('/content/df_normalized.csv')
dfComplete
```

Out[5]:

| | IATA code | Destination | Flight | Airline | Status | Origin | temperature_2m | relative_humidity_2m | de |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.248447 | 0.262821 | 0.706916 | 0.662338 | 0.0 | 0.444444 | 0.425620 | 0.662651 | |
| **1** | 0.248447 | 0.262821 | 0.458239 | 0.350649 | 0.0 | 0.444444 | 0.425620 | 0.662651 | |
| **2** | 0.248447 | 0.262821 | 0.829743 | 0.766234 | 0.0 | 0.444444 | 0.425620 | 0.662651 | |
| **3** | 0.248447 | 0.262821 | 0.526266 | 0.448052 | 0.0 | 0.111111 | 0.421488 | 0.710843 | |
| **4** | 0.248447 | 0.262821 | 0.702381 | 0.649351 | 0.0 | 0.111111 | 0.421488 | 0.710843 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **102185** | 0.956522 | 0.160256 | 0.097317 | 0.207792 | 0.0 | 0.888889 | 0.595041 | 0.686747 | |
| **102186** | 0.956522 | 0.160256 | 0.935941 | 0.889610 | 0.0 | 0.888889 | 0.595041 | 0.686747 | |
| **102187** | 0.347826 | 0.794872 | 0.737717 | 0.662338 | 0.0 | 0.888889 | 0.599174 | 0.650602 | |
| **102188** | 0.347826 | 0.794872 | 0.448602 | 0.350649 | 0.0 | 0.888889 | 0.599174 | 0.650602 | |
| **102189** | 0.347826 | 0.794872 | 0.801965 | 0.720779 | 0.0 | 0.888889 | 0.599174 | 0.650602 | |

102190 rows × 24 columns

## Passo 1: Escolher a coluna Status para predição (classificação)

In [6]:
```
X = dfComplete.drop(columns=['Status'])
y = dfComplete['Status']
```

## Passo 2: Separar os dados em treinamento, validação e teste

In [7]:
```
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_st
```

In [8]:
```
from imblearn.over_sampling import RandomOverSampler

# Verifique a distribuição de classes antes do oversampling
print("Antes do oversampling:")
print(pd.Series(y_train).value_counts())

# Aplicar oversampling
oversampler = RandomOverSampler(random_state=42)
X_train, y_train = oversampler.fit_resample(X_train, y_train)

# Verifique a distribuição de classes após o oversampling
print("\nApós o oversampling:")
print(pd.Series(y_train).value_counts())
```

```
Antes do oversampling:
Status
0.0    68298
1.0     3235
Name: count, dtype: int64

Após o oversampling:
Status
0.0    68298
1.0    68298
Name: count, dtype: int64
```

In [9]:
```
print("Antes do oversampling:")
print(pd.Series(y_val).value_counts())

oversampler = RandomOverSampler(random_state=42)
X_val, y_val = oversampler.fit_resample(X_val, y_val)

print("\nApós o oversampling:")
print(pd.Series(y_val).value_counts())
```

```
Antes do oversampling:
Status
0.0    14635
1.0      693
Name: count, dtype: int64

Após o oversampling:
Status
0.0    14635
1.0    14635
Name: count, dtype: int64
```

## Passo 3: Selecionar 4 algoritmos

In [ ]:
```
models = {
  'Logistic Regression': LogisticRegression(),
  'Decision Tree': DecisionTreeClassifier(),
  'Random Forest': RandomForestClassifier(),
  'Gradient Boosting': GradientBoostingClassifier()
}
```

## Passo 4: Adicionar MLFlow no treinamento dos modelos para rastreamento

In [ ]:
```
mlflow.set_experiment("Model Selection Experiment")
```

```
2024/07/12 12:31:53 INFO mlflow.tracking.fluent: Experiment with name 'Model Selection E
xperiment' does not exist. Creating a new experiment.
```

Out[ ]:
```
<Experiment: artifact_location='file:///content/mlruns/757061576949493941', creation_tim
e=1720787513268, experiment_id='757061576949493941', last_update_time=1720787513268, lif
ecycle_stage='active', name='Model Selection Experiment', tags={}>
```

## Passo 5: Executar uma ferramenta de seleção de hiper-parâmetros

In [ ]:
```
best_models = {}
best_scores = {}

def objective(trial, model_name):
  if model_name == 'Logistic Regression':
    C = trial.suggest_categorical('C', [0.01, 0.1, 1, 10, 100])
    model = LogisticRegression(C=C, random_state=42)
  elif model_name == 'Decision Tree':
    max_depth = trial.suggest_int('max_depth', 3, 15)
    min_samples_split = trial.suggest_int('min_samples_split', 2, 20)
    min_samples_leaf = trial.suggest_int('min_samples_leaf', 1, 10)
    model = DecisionTreeClassifier( max_depth=max_depth, min_samples_split=min_samples_s
  elif model_name == 'Random Forest':
    n_estimators = trial.suggest_int('n_estimators', 10, 100)
    max_depth = trial.suggest_categorical('max_depth', [None, 10, 20, 30])
    model = RandomForestClassifier(n_estimators=n_estimators, max_depth=max_depth, rando
  elif model_name == 'Gradient Boosting':
    n_estimators = trial.suggest_int('n_estimators', 50, 200)
    learning_rate = trial.suggest_float('learning_rate', 0.01, 0.2)
    model = GradientBoostingClassifier(n_estimators=n_estimators, learning_rate=learning

  model.fit(X_train, y_train)
  score = model.score(X_val, y_val)
  return score
```

In [ ]:
```
for model_name in models.keys():
  study = optuna.create_study(direction='maximize')
  study.optimize(lambda trial: objective(trial, model_name), n_trials=10)

  best_model = models[model_name].set_params(**study.best_params)
  best_model.fit(X_train, y_train)
  best_score = best_model.score(X_val, y_val)

  with mlflow.start_run(run_name=model_name):
    mlflow.log_params(study.best_params)
    mlflow.log_metric("accuracy", best_score)
    mlflow.sklearn.log_model(best_model, "model")

  best_models[model_name] = best_model
  best_scores[model_name] = best_score

print("Best models and their scores:")
print(best_scores)
```

```
[I 2024-07-12 12:31:55,431] A new study created in memory with name: no-name-8f7dd9ca-e7
8c-4d88-9554-9d2eab07f735
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Convergen
ceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
[I 2024-07-12 12:31:58,521] Trial 0 finished with value: 0.5645370686709942 and paramete
rs: {'C': 1}. Best is trial 0 with value: 0.5645370686709942.
[I 2024-07-12 12:32:01,392] Trial 1 finished with value: 0.5590365562008883 and paramete
rs: {'C': 0.01}. Best is trial 0 with value: 0.5645370686709942.
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Convergen
ceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
[I 2024-07-12 12:32:06,605] Trial 2 finished with value: 0.5593098735907072 and paramete
rs: {'C': 0.1}. Best is trial 0 with value: 0.5645370686709942.
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Convergen
ceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
[I 2024-07-12 12:32:11,832] Trial 3 finished with value: 0.5645370686709942 and paramete
rs: {'C': 1}. Best is trial 0 with value: 0.5645370686709942.
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Convergen
ceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
[I 2024-07-12 12:32:18,000] Trial 4 finished with value: 0.5675777246327297 and paramete
rs: {'C': 10}. Best is trial 4 with value: 0.5675777246327297.
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Convergen
ceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
[I 2024-07-12 12:32:22,331] Trial 5 finished with value: 0.5659378202938162 and paramete
rs: {'C': 100}. Best is trial 4 with value: 0.5675777246327297.
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Convergen
```

```
ceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
[I 2024-07-12 12:32:27,416] Trial 6 finished with value: 0.5675777246327297 and paramete
rs: {'C': 10}. Best is trial 4 with value: 0.5675777246327297.
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Convergen
ceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
[I 2024-07-12 12:32:34,040] Trial 7 finished with value: 0.5593098735907072 and paramete
rs: {'C': 0.1}. Best is trial 4 with value: 0.5675777246327297.
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Convergen
ceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
[I 2024-07-12 12:32:39,215] Trial 8 finished with value: 0.5659378202938162 and paramete
rs: {'C': 100}. Best is trial 4 with value: 0.5675777246327297.
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Convergen
ceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
[I 2024-07-12 12:32:43,931] Trial 9 finished with value: 0.5593098735907072 and paramete
rs: {'C': 0.1}. Best is trial 4 with value: 0.5675777246327297.
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: Convergen
ceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/_distutils_hack/__init__.py:33: UserWarning: Set
uptools is replacing distutils.
  warnings.warn("Setuptools is replacing distutils.")
[I 2024-07-12 12:32:56,622] A new study created in memory with name: no-name-7d7c7a90-99
43-48de-aa63-5da3ecb60bc5
[I 2024-07-12 12:33:01,121] Trial 0 finished with value: 0.8208062862999659 and paramete
rs: {'max_depth': 13, 'min_samples_split': 5, 'min_samples_leaf': 3}. Best is trial 0 wi
th value: 0.8208062862999659.
[I 2024-07-12 12:33:02,856] Trial 1 finished with value: 0.7707892039631021 and paramete
rs: {'max_depth': 9, 'min_samples_split': 20, 'min_samples_leaf': 3}. Best is trial 0 wi
th value: 0.8208062862999659.
[I 2024-07-12 12:33:05,058] Trial 2 finished with value: 0.8172189955585925 and paramete
rs: {'max_depth': 12, 'min_samples_split': 20, 'min_samples_leaf': 9}. Best is trial 0 w
ith value: 0.8208062862999659.
```

```
[I 2024-07-12 12:33:07,216] Trial 3 finished with value: 0.8031431499829177 and paramete
rs: {'max_depth': 11, 'min_samples_split': 3, 'min_samples_leaf': 6}. Best is trial 0 wi
th value: 0.8208062862999659.
[I 2024-07-12 12:33:08,315] Trial 4 finished with value: 0.6903313973351555 and paramete
rs: {'max_depth': 5, 'min_samples_split': 2, 'min_samples_leaf': 1}. Best is trial 0 wit
h value: 0.8208062862999659.
[I 2024-07-12 12:33:11,018] Trial 5 finished with value: 0.818209771096686 and parameter
s: {'max_depth': 12, 'min_samples_split': 16, 'min_samples_leaf': 6}. Best is trial 0 wi
th value: 0.8208062862999659.
[I 2024-07-12 12:33:13,300] Trial 6 finished with value: 0.8322856166723608 and paramete
rs: {'max_depth': 15, 'min_samples_split': 8, 'min_samples_leaf': 6}. Best is trial 6 wi
th value: 0.8322856166723608.
[I 2024-07-12 12:33:13,964] Trial 7 finished with value: 0.6328322514519986 and paramete
rs: {'max_depth': 3, 'min_samples_split': 5, 'min_samples_leaf': 1}. Best is trial 6 wit
h value: 0.8322856166723608.
[I 2024-07-12 12:33:15,739] Trial 8 finished with value: 0.8151349504612231 and paramete
rs: {'max_depth': 12, 'min_samples_split': 5, 'min_samples_leaf': 1}. Best is trial 6 wi
th value: 0.8322856166723608.
[I 2024-07-12 12:33:17,265] Trial 9 finished with value: 0.8180389477280492 and paramete
rs: {'max_depth': 13, 'min_samples_split': 19, 'min_samples_leaf': 8}. Best is trial 6 w
ith value: 0.8322856166723608.
[I 2024-07-12 12:33:21,273] A new study created in memory with name: no-name-631f6812-24
7d-4080-80c2-af6d81288f09
[I 2024-07-12 12:33:26,944] Trial 0 finished with value: 0.8062521352921079 and paramete
rs: {'n_estimators': 25, 'max_depth': 10}. Best is trial 0 with value: 0.806252135292107
9.
[I 2024-07-12 12:33:38,048] Trial 1 finished with value: 0.8256918346429791 and paramete
rs: {'n_estimators': 40, 'max_depth': 20}. Best is trial 1 with value: 0.825691834642979
1.
[I 2024-07-12 12:33:44,176] Trial 2 finished with value: 0.8290741373419884 and paramete
rs: {'n_estimators': 18, 'max_depth': 20}. Best is trial 2 with value: 0.829074137341988
4.
[I 2024-07-12 12:33:56,985] Trial 3 finished with value: 0.809156132558934 and parameter
s: {'n_estimators': 42, 'max_depth': 30}. Best is trial 2 with value: 0.829074137341988
4.
[I 2024-07-12 12:34:02,732] Trial 4 finished with value: 0.8075845575674753 and paramete
rs: {'n_estimators': 21, 'max_depth': 30}. Best is trial 2 with value: 0.829074137341988
4.
[I 2024-07-12 12:34:26,246] Trial 5 finished with value: 0.8004441407584557 and paramete
rs: {'n_estimators': 72, 'max_depth': None}. Best is trial 2 with value: 0.8290741373419
884.
[I 2024-07-12 12:34:44,353] Trial 6 finished with value: 0.8172531602323198 and paramete
rs: {'n_estimators': 82, 'max_depth': 10}. Best is trial 2 with value: 0.829074137341988
4.
[I 2024-07-12 12:34:53,818] Trial 7 finished with value: 0.8128459173214896 and paramete
rs: {'n_estimators': 30, 'max_depth': 10}. Best is trial 2 with value: 0.829074137341988
4.
[I 2024-07-12 12:35:12,926] Trial 8 finished with value: 0.8166040314314998 and paramete
rs: {'n_estimators': 76, 'max_depth': 10}. Best is trial 2 with value: 0.829074137341988
4.
[I 2024-07-12 12:35:19,851] Trial 9 finished with value: 0.7966518619747182 and paramete
rs: {'n_estimators': 16, 'max_depth': 30}. Best is trial 2 with value: 0.829074137341988
4.
[I 2024-07-12 12:35:29,109] A new study created in memory with name: no-name-78d17e43-2b
f8-44ef-bc66-dacecd5ea7b0
[I 2024-07-12 12:36:03,973] Trial 0 finished with value: 0.7680218653911856 and paramete
rs: {'n_estimators': 63, 'learning_rate': 0.1951750869505307}. Best is trial 0 with valu
e: 0.7680218653911856.
[I 2024-07-12 12:36:33,650] Trial 1 finished with value: 0.7092927912538435 and paramete
rs: {'n_estimators': 56, 'learning_rate': 0.04049496896945324}. Best is trial 0 with val
ue: 0.7680218653911856.
[I 2024-07-12 12:37:38,016] Trial 2 finished with value: 0.7237102835667919 and paramete
rs: {'n_estimators': 120, 'learning_rate': 0.03133128604148498}. Best is trial 0 with va
lue: 0.7680218653911856.
[I 2024-07-12 12:38:25,809] Trial 3 finished with value: 0.7694567816877349 and paramete
rs: {'n_estimators': 91, 'learning_rate': 0.11850200879903874}. Best is trial 3 with val
```

```
ue: 0.7694567816877349.
[I 2024-07-12 12:39:46,645] Trial 4 finished with value: 0.7757430816535702 and paramete
rs: {'n_estimators': 158, 'learning_rate': 0.08205147484302205}. Best is trial 4 with va
lue: 0.7757430816535702.
[I 2024-07-12 12:41:12,500] Trial 5 finished with value: 0.7488896481038606 and paramete
rs: {'n_estimators': 170, 'learning_rate': 0.038879831719854666}. Best is trial 4 with v
alue: 0.7757430816535702.
[I 2024-07-12 12:42:32,032] Trial 6 finished with value: 0.775913905022207 and parameter
s: {'n_estimators': 149, 'learning_rate': 0.09197552096755536}. Best is trial 6 with val
ue: 0.775913905022207.
[I 2024-07-12 12:43:06,292] Trial 7 finished with value: 0.7091902972326615 and paramete
rs: {'n_estimators': 64, 'learning_rate': 0.03784344590559623}. Best is trial 6 with val
ue: 0.775913905022207.
[I 2024-07-12 12:43:39,996] Trial 8 finished with value: 0.7061154765971985 and paramete
rs: {'n_estimators': 63, 'learning_rate': 0.03005173265541828}. Best is trial 6 with val
ue: 0.775913905022207.
[I 2024-07-12 12:44:25,790] Trial 9 finished with value: 0.7709258626580117 and paramete
rs: {'n_estimators': 86, 'learning_rate': 0.13995364932321683}. Best is trial 6 with val
ue: 0.775913905022207.
Best models and their scores:
{'Logistic Regression': 0.5675777246327297, 'Decision Tree': 0.8339596856850017, 'Random
Forest': 0.8157157499145883, 'Gradient Boosting': 0.7759480696959344}
```

### 5.2 Selecionar o modelo com melhor resultado na métrica de avaliação

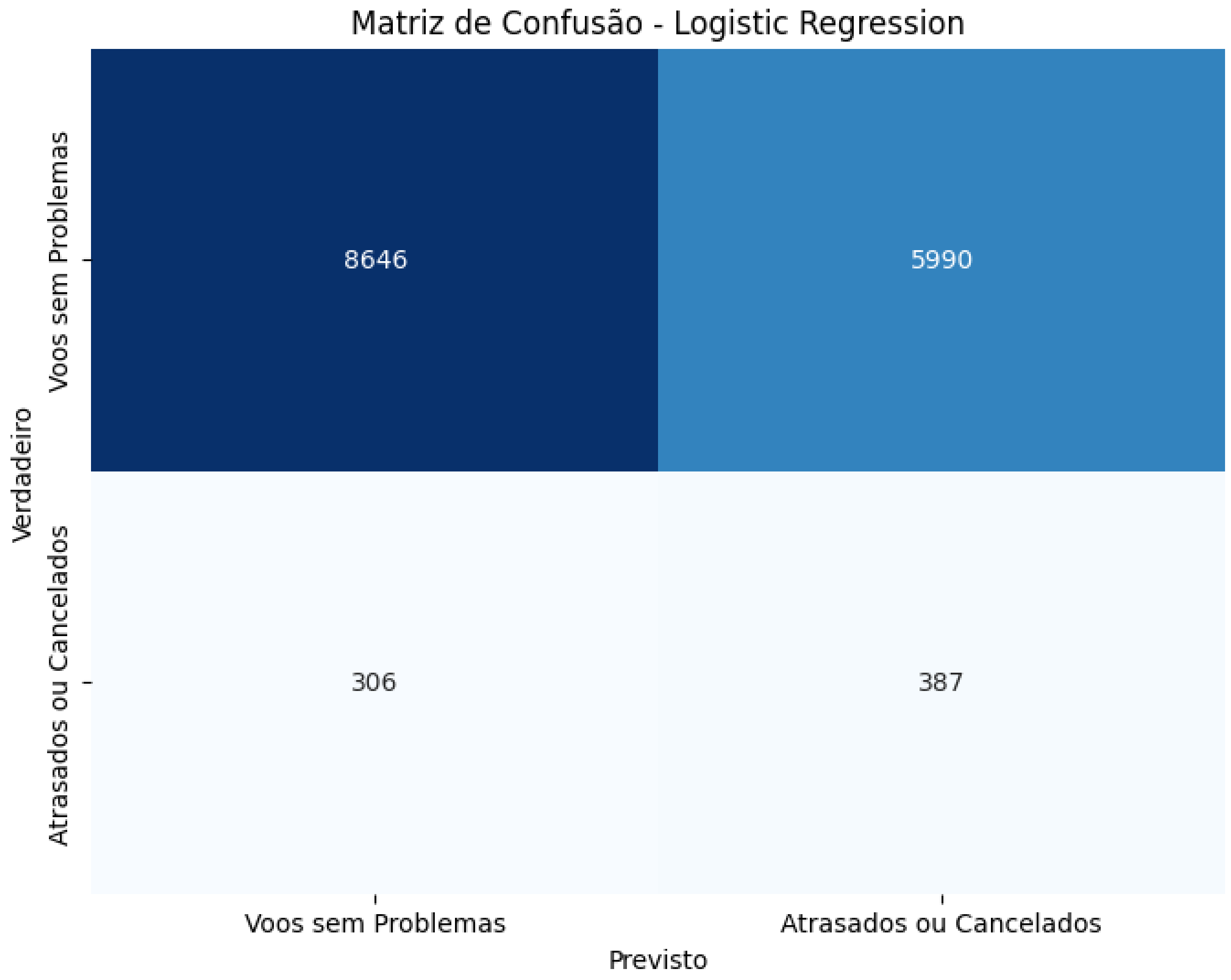
In [ ]:
```
best_model_name = max(best_scores, key=best_scores.get)
best_model = best_models[best_model_name]
print(f"Melhor modelo: {best_model_name}")
```

```
Melhor modelo: Decision Tree
```

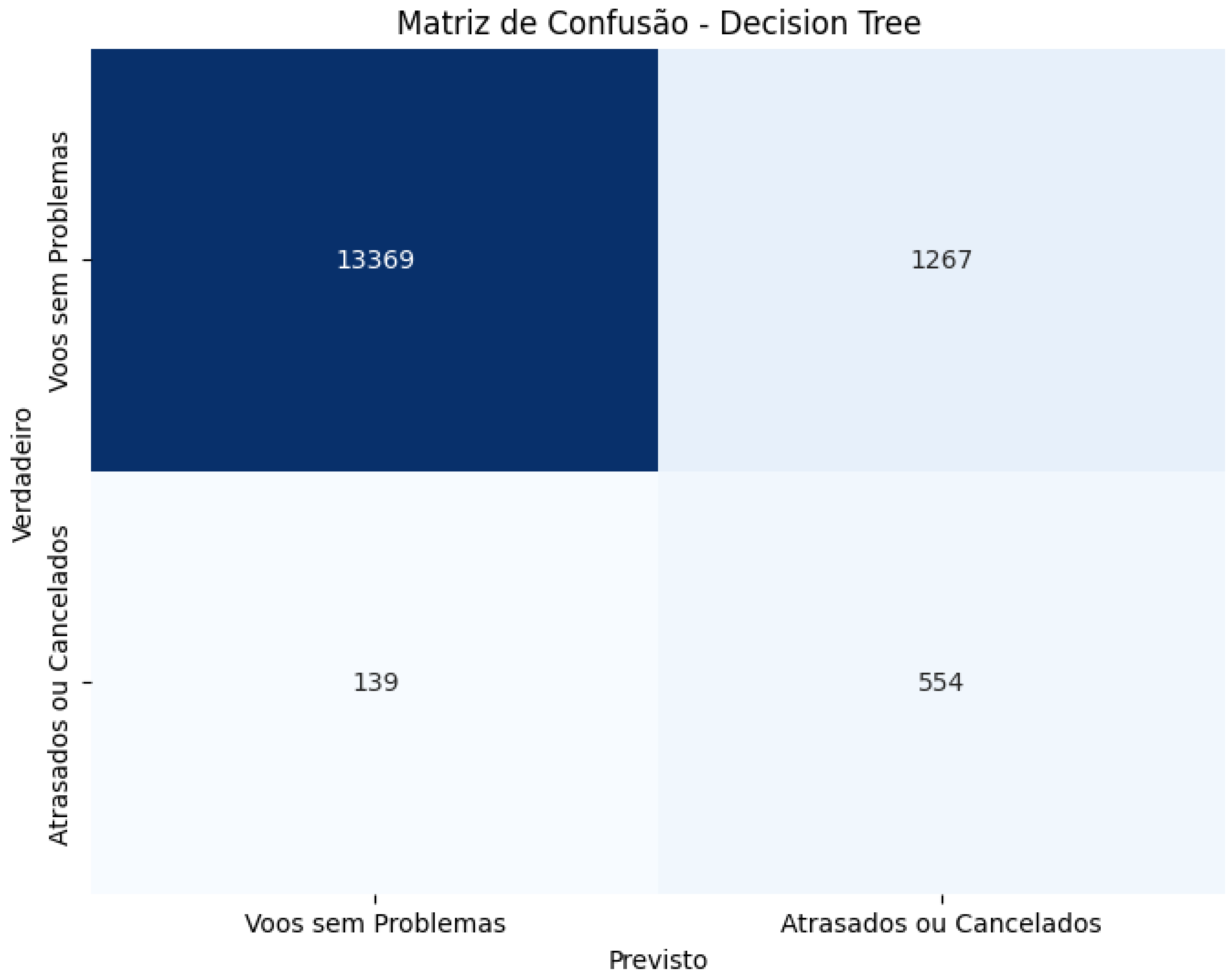
### 5.3 Executar o melhor modelo de cada algoritmo no conjunto de teste

In [ ]:
```
for model_name in models.keys():
  y_pred = best_models[model_name].predict(X_test)
  test_accuracy = accuracy_score(y_test, y_pred)

  print(f"Melhor modelo: {model_name}")
  print(f"Acurácia no conjunto de teste: {test_accuracy}")
  print(classification_report(y_test, y_pred))

  cm = confusion_matrix(y_test, y_pred)
  plt.figure(figsize=(8, 6))
  sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
              xticklabels=['Voos sem Problemas', 'Atrasados ou Cancelados'], yticklabels
  plt.xlabel('Previsto')
  plt.ylabel('Verdadeiro')
  plt.title(f'Matriz de Confusão - {model_name}')
  plt.show()
```

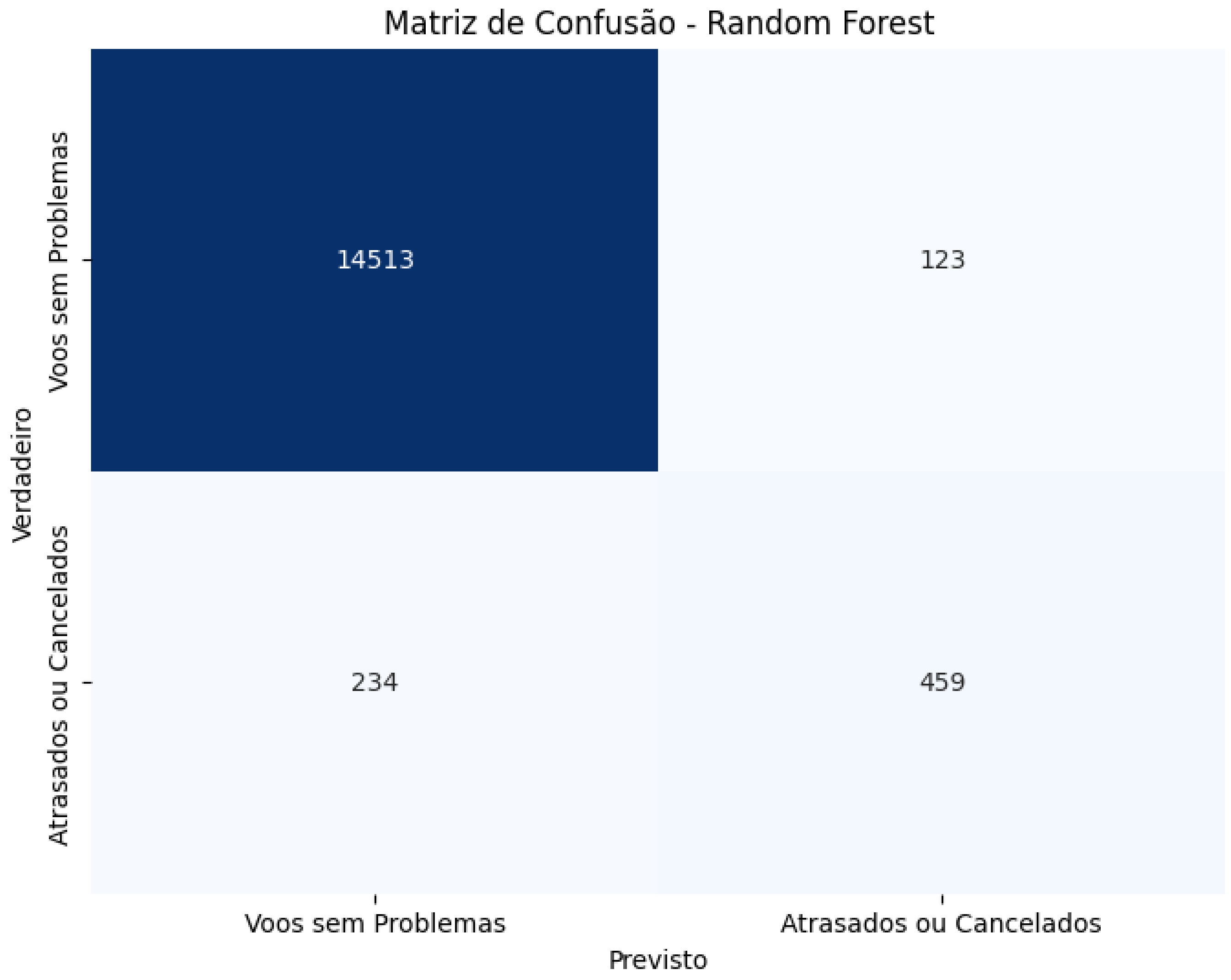
```
Melhor modelo: Logistic Regression
Acurácia no conjunto de teste: 0.589275229956292
              precision    recall  f1-score   support

         0.0       0.97      0.59      0.73     14636
         1.0       0.06      0.56      0.11       693

    accuracy                           0.59     15329
   macro avg       0.51      0.57      0.42     15329
weighted avg       0.92      0.59      0.70     15329
```



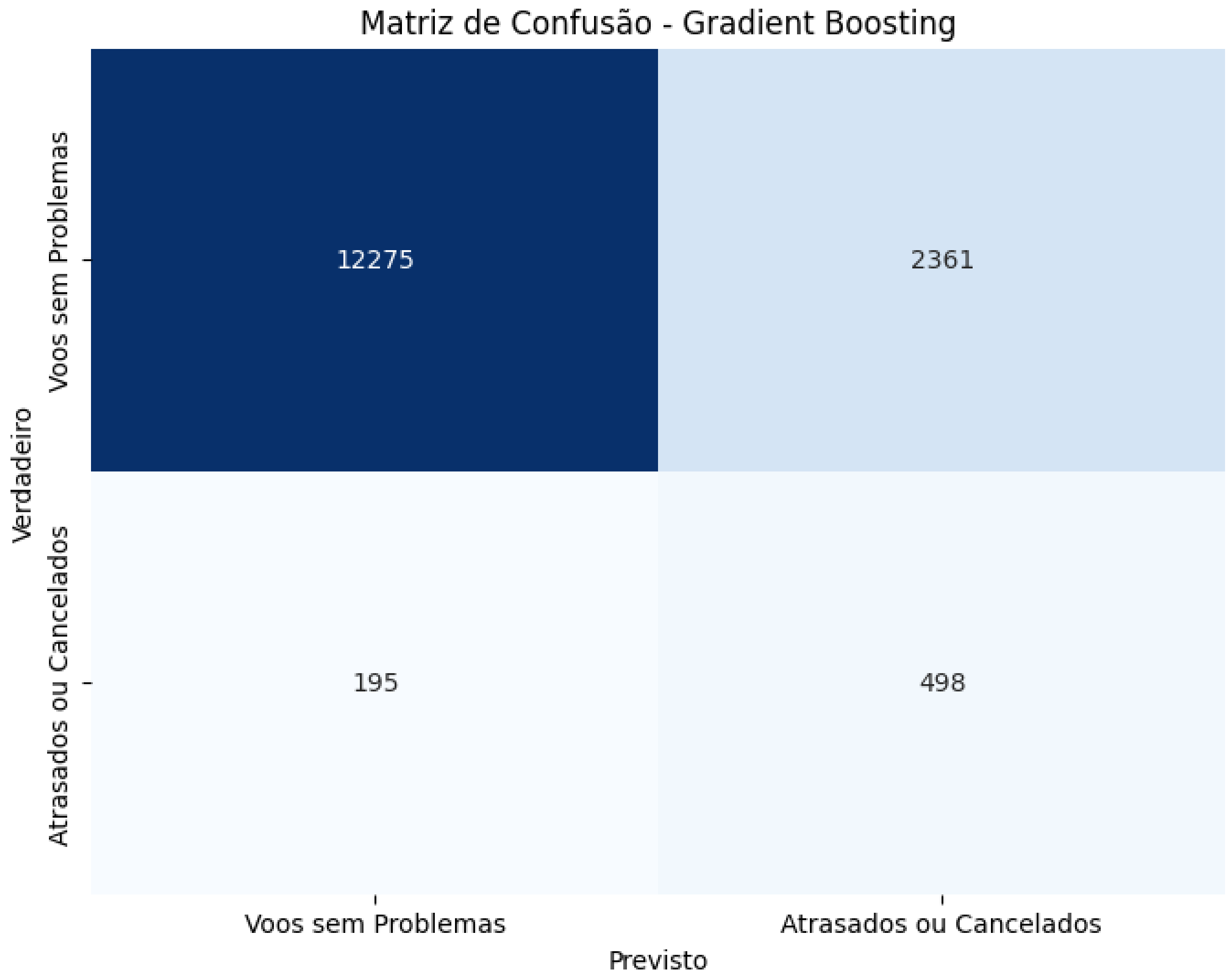
```
Melhor modelo: Decision Tree
Acurácia no conjunto de teste: 0.9082784265118403
              precision    recall  f1-score   support

         0.0       0.99      0.91      0.95     14636
         1.0       0.30      0.80      0.44       693

    accuracy                           0.91     15329
   macro avg       0.65      0.86      0.70     15329
weighted avg       0.96      0.91      0.93     15329
```



```
Melhor modelo: Random Forest
Acurácia no conjunto de teste: 0.9767108095766195
              precision    recall  f1-score   support

         0.0       0.98      0.99      0.99     14636
         1.0       0.79      0.66      0.72       693

    accuracy                           0.98     15329
   macro avg       0.89      0.83      0.85     15329
weighted avg       0.98      0.98      0.98     15329
```



```
Melhor modelo: Gradient Boosting
Acurácia no conjunto de teste: 0.8332572248678974
              precision    recall  f1-score   support

         0.0       0.98      0.84      0.91     14636
         1.0       0.17      0.72      0.28       693

    accuracy                           0.83     15329
   macro avg       0.58      0.78      0.59     15329
weighted avg       0.95      0.83      0.88     15329
```



```
In [ ]: !zip -r /content/mlruns.zip /content/mlruns
```

```
  adding: content/mlruns/ (stored 0%)
  adding: content/mlruns/.trash/ (stored 0%)
  adding: content/mlruns/757061576949493941/ (stored 0%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/ (stored
0%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/params/ (st
ored 0%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/params/lear
ning_rate (stored 0%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/params/n_es
timators (stored 0%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/tags/ (stor
ed 0%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/tags/mlflo
w.user (stored 0%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/tags/mlflo
w.source.name (deflated 5%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/tags/mlflo
w.log-model.history (deflated 44%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/tags/mlflo
w.source.type (stored 0%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/tags/mlflo
w.runName (stored 0%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/meta.yaml
(deflated 44%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/artifacts/
(stored 0%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/artifacts/m
```

```
odel/ (stored 0%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/artifacts/m
odel/python_env.yaml (deflated 18%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/artifacts/m
odel/MLmodel (deflated 43%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/artifacts/m
odel/conda.yaml (deflated 33%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/artifacts/m
odel/requirements.txt (deflated 18%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/artifacts/m
odel/model.pkl (deflated 68%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/metrics/ (s
tored 0%)
  adding: content/mlruns/757061576949493941/abd3d0cf75544fbebfd4bed44984c709/metrics/acc
uracy (stored 0%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/ (stored
0%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/params/ (st
ored 0%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/params/max_
depth (stored 0%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/params/n_es
timators (stored 0%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/tags/ (stor
ed 0%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/tags/mlflo
w.user (stored 0%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/tags/mlflo
w.source.name (deflated 5%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/tags/mlflo
w.log-model.history (deflated 44%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/tags/mlflo
w.source.type (stored 0%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/tags/mlflo
w.runName (stored 0%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/meta.yaml
(deflated 45%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/artifacts/
(stored 0%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/artifacts/m
odel/ (stored 0%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/artifacts/m
odel/python_env.yaml (deflated 18%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/artifacts/m
odel/MLmodel (deflated 43%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/artifacts/m
odel/conda.yaml (deflated 33%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/artifacts/m
odel/requirements.txt (deflated 18%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/artifacts/m
odel/model.pkl (deflated 76%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/metrics/ (s
tored 0%)
  adding: content/mlruns/757061576949493941/a2c18b0733c8499092f17e8b8561c791/metrics/acc
uracy (stored 0%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/ (stored
0%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/params/ (st
ored 0%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/params/C (s
tored 0%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/tags/ (stor
ed 0%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/tags/mlflo
w.user (stored 0%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/tags/mlflo
```

```
w.source.name (deflated 5%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/tags/mlflo
w.log-model.history (deflated 44%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/tags/mlflo
w.source.type (stored 0%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/tags/mlflo
w.runName (stored 0%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/meta.yaml
(deflated 45%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/artifacts/
(stored 0%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/artifacts/m
odel/ (stored 0%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/artifacts/m
odel/python_env.yaml (deflated 18%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/artifacts/m
odel/MLmodel (deflated 43%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/artifacts/m
odel/conda.yaml (deflated 33%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/artifacts/m
odel/requirements.txt (deflated 18%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/artifacts/m
odel/model.pkl (deflated 32%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/metrics/ (s
tored 0%)
  adding: content/mlruns/757061576949493941/f5f3c50b661d4fe19b797a394270ccf9/metrics/acc
uracy (stored 0%)
  adding: content/mlruns/757061576949493941/meta.yaml (deflated 32%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/ (stored
0%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/params/ (st
ored 0%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/params/min_
samples_leaf (stored 0%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/params/max_
depth (stored 0%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/params/min_
samples_split (stored 0%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/tags/ (stor
ed 0%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/tags/mlflo
w.user (stored 0%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/tags/mlflo
w.source.name (deflated 5%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/tags/mlflo
w.log-model.history (deflated 43%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/tags/mlflo
w.source.type (stored 0%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/tags/mlflo
w.runName (stored 0%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/meta.yaml
(deflated 45%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/artifacts/
(stored 0%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/artifacts/m
odel/ (stored 0%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/artifacts/m
odel/python_env.yaml (deflated 18%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/artifacts/m
odel/MLmodel (deflated 43%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/artifacts/m
odel/conda.yaml (deflated 33%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/artifacts/m
odel/requirements.txt (deflated 18%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/artifacts/m
odel/model.pkl (deflated 75%)
```

```
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/metrics/ (s
tored 0%)
  adding: content/mlruns/757061576949493941/821283f9b02e4778b80412cc858b0df0/metrics/acc
uracy (stored 0%)
  adding: content/mlruns/0/ (stored 0%)
  adding: content/mlruns/0/meta.yaml (deflated 24%)
```

In [ ]:
```python
from google.colab import files
files.download('/content/mlruns.zip')
```

## Passo 6: Realizar diagnóstico do melhor modelo geral e melhorá-lo

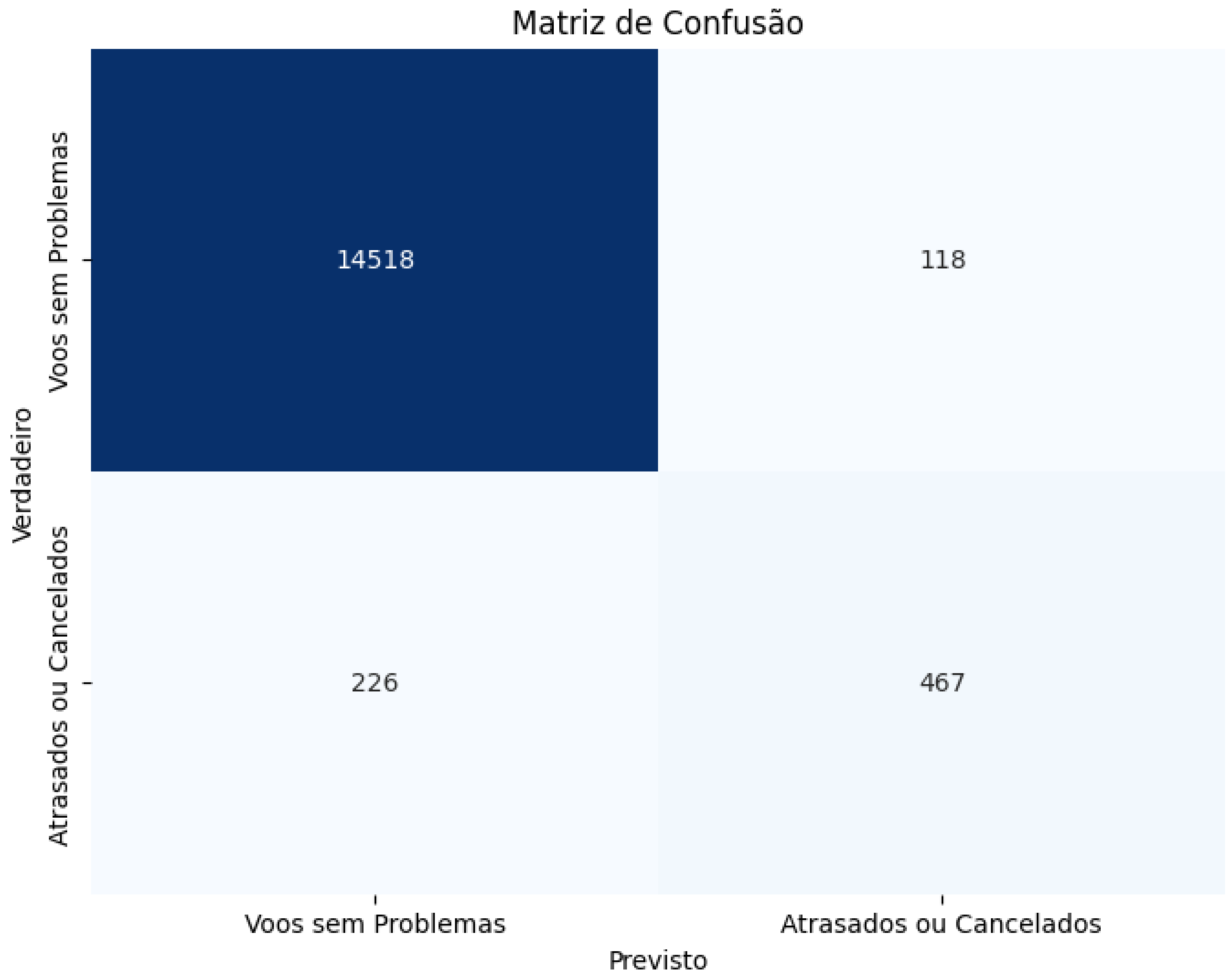
O melhor modelo de acordo com o conjunto de teste foi a Random Forest com n_estimators igual a 18 e max_depth igual a 20.

In [10]:
```python
model = RandomForestClassifier(n_estimators=18, max_depth=20, random_state=42)
model.fit(X_train, y_train)
```

Out[10]:
```
RandomForestClassifier

RandomForestClassifier(max_depth=20, n_estimators=18, random_state=42)
```

In [13]:
```python
y_pred = model.predict(X_test)
test_accuracy = accuracy_score(y_test, y_pred)

print(f"Acurácia no conjunto de teste: {test_accuracy}")
print(classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Voos sem Problemas', 'Atrasados ou Cancelados'], yticklabels=[
plt.xlabel('Previsto')
plt.ylabel('Verdadeiro')
plt.title(f'Matriz de Confusão')
plt.show()
```

```
Acurácia no conjunto de teste: 0.9775588753343336
              precision    recall  f1-score   support

         0.0       0.98      0.99      0.99     14636
         1.0       0.80      0.67      0.73       693

    accuracy                           0.98     15329
   macro avg       0.89      0.83      0.86     15329
weighted avg       0.98      0.98      0.98     15329
```



O conjunto de teste está desbalanceado, então a acurácia não é uma métrica tão boa. Iremos avaliar a quantidade de instância classificadas erroneamente. Nesse caso, 118+226=344.

In [14]:
```
feature_importances = model.feature_importances_
features = X.columns
feature_importance_list = list(zip(features, feature_importances))
feature_importance_list = sorted(feature_importance_list, key=lambda x: x[1], reverse=Tr

for feature, importance in feature_importance_list:
    print(f'Feature: {feature}, Importance: {importance}')
```

```
Feature: dew_point_2m_dst, Importance: 0.06762733101253597
Feature: IATA code, Importance: 0.06292133504454783
Feature: relative_humidity_2m, Importance: 0.06204796330223724
Feature: wind_direction_10m, Importance: 0.059396939220753536
Feature: pressure_msl, Importance: 0.05784860856146626
Feature: dew_point_2m, Importance: 0.05600355416308637
Feature: Origin, Importance: 0.05597972767971038
Feature: temperature_2m, Importance: 0.054977755867566175
Feature: wind_speed_10m, Importance: 0.052941613952188604
Feature: pressure_msl_dst, Importance: 0.052835601799942
Feature: temperature_2m_dst, Importance: 0.05190383937793102
Feature: Flight, Importance: 0.05033096160420999
Feature: wind_direction_10m_dst, Importance: 0.04979836110707678
Feature: wind_speed_10m_dst, Importance: 0.04762134931726888
Feature: Destination, Importance: 0.043354986314556074
Feature: relative_humidity_2m_dst, Importance: 0.0423337948564907
Feature: cloud_cover, Importance: 0.04206443806703302
Feature: cloud_cover_dst, Importance: 0.03705385727973402
Feature: Airline, Importance: 0.031069707137325393
```

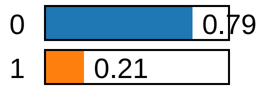
```
Feature: precipitation_dst, Importance: 0.008875781826125188
Feature: precipitation, Importance: 0.008580258919092042
Feature: outlier, Importance: 0.004432233589122546
Feature: snowfall_dst, Importance: 0.0
```

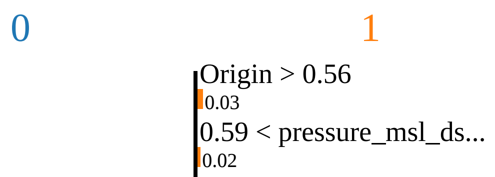
In [34]:
```
# Interpretação local com LIME
explainer = lime.lime_tabular.LimeTabularExplainer(X_train.values, feature_names=X_train
idxs = [random.randint(0, len(X_test)-1) for _ in range(30)]

for idx in idxs:
  print(f"Índice: {idx}")
  exp = explainer.explain_instance(X_test.values[idx], model.predict_proba, num_features
  exp.show_in_notebook(show_table=True)
```

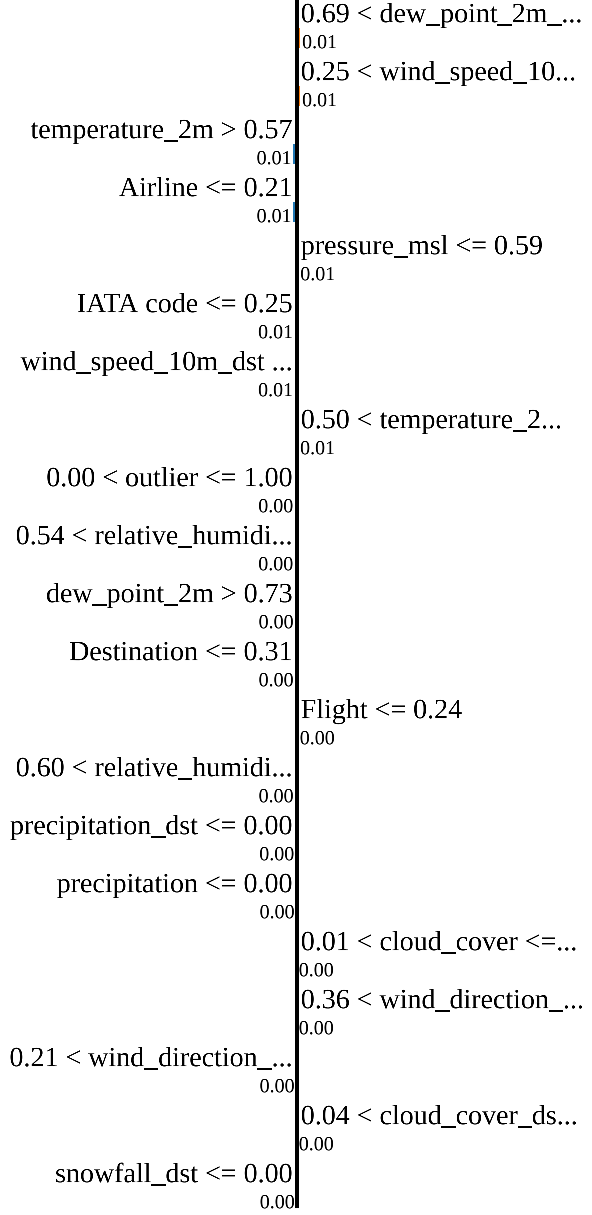
```
Índice: 13745
Intercept 0.1006720194151927
Prediction_local [0.12532969]
Right: 0.20885104490236397
```

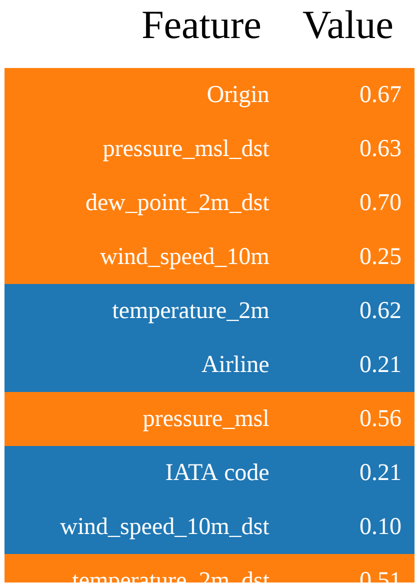
```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

Prediction probabilities





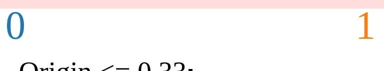


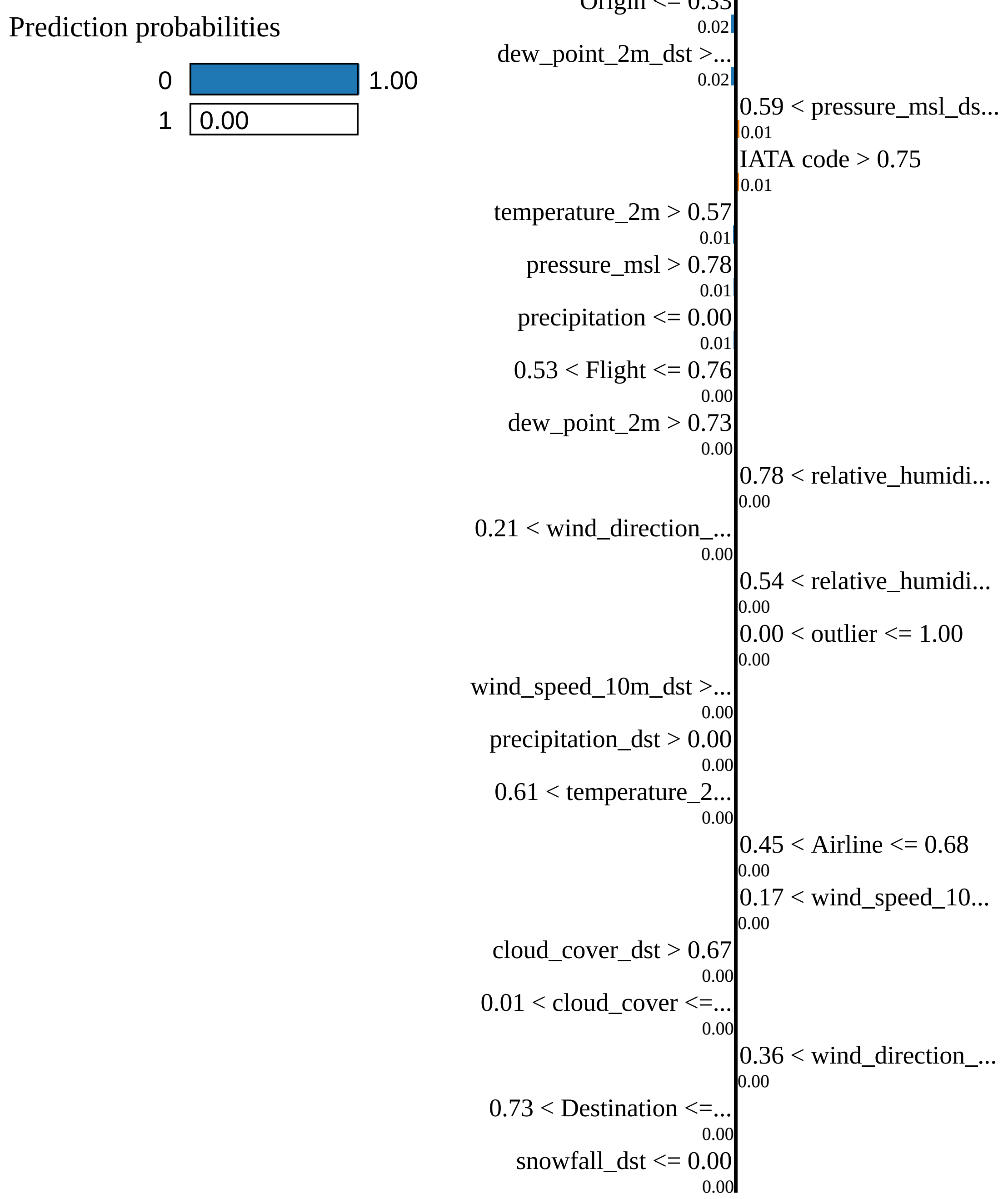


| Feature | Value |
|---|---|
| Origin | 0.67 |
| pressure_msl_dst | 0.63 |
| dew_point_2m_dst | 0.70 |
| wind_speed_10m | 0.25 |
| temperature_2m | 0.62 |
| Airline | 0.21 |
| pressure_msl | 0.56 |
| IATA code | 0.21 |
| wind_speed_10m_dst | 0.10 |
| temperature_2m_dst | 0.51 |

```
Índice: 7
Intercept 0.11715217689643048
Prediction_local [0.07504593]
Right: 0.0
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

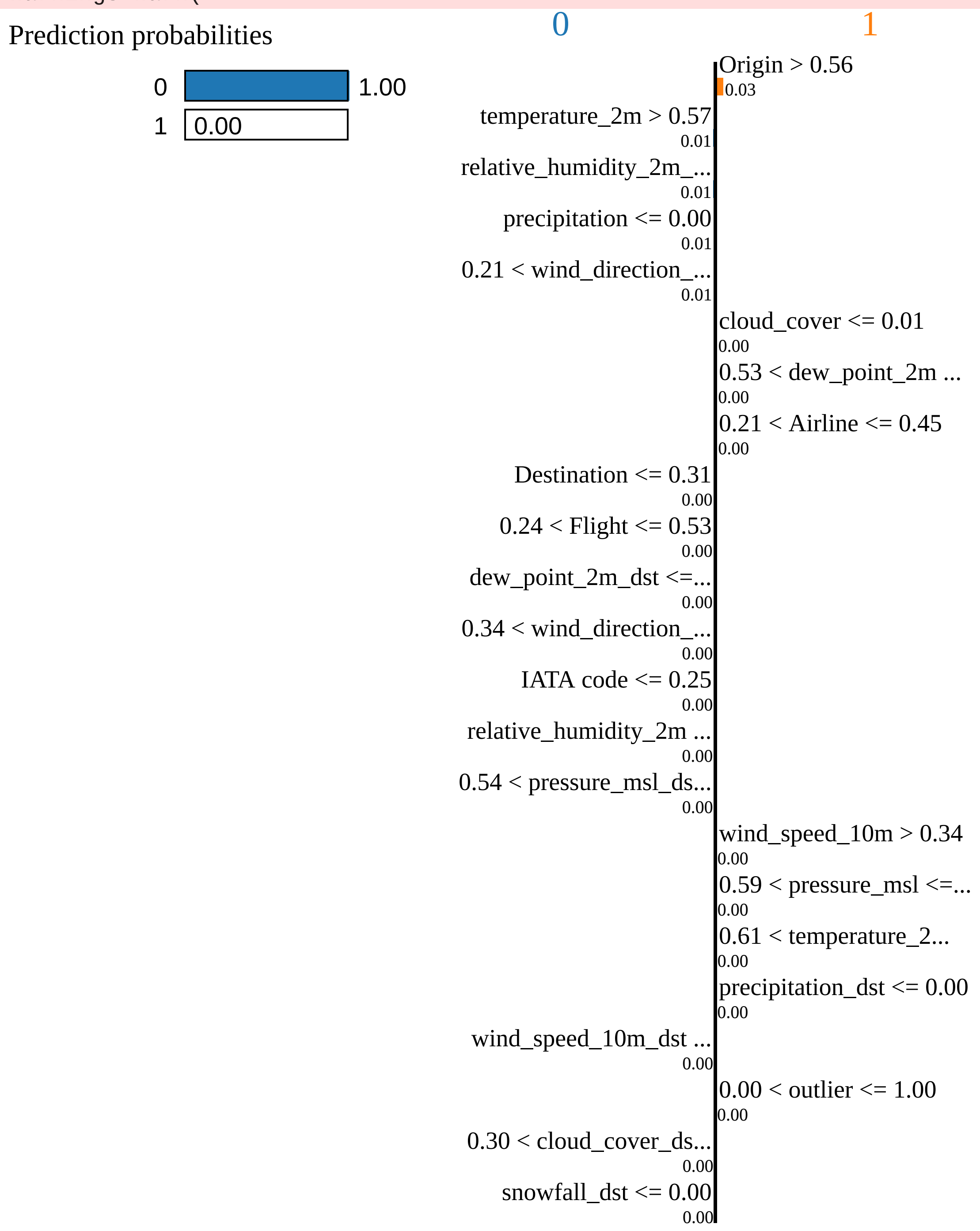




| Feature | Value |
|---|---|
| Origin | 0.33 |
| dew_point_2m_dst | 0.92 |
| pressure_msl_dst | 0.60 |
| IATA code | 0.89 |
| temperature_2m | 0.65 |
| pressure_msl | 0.81 |
| precipitation | 0.00 |
| Flight | 0.58 |
| dew_point_2m | 0.80 |
| relative_humidity_2m_dst | 0.87 |

```
Índice: 10494
Intercept 0.10407345882309971
Prediction_local [0.10592171]
Right: 0.0
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
```

```
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

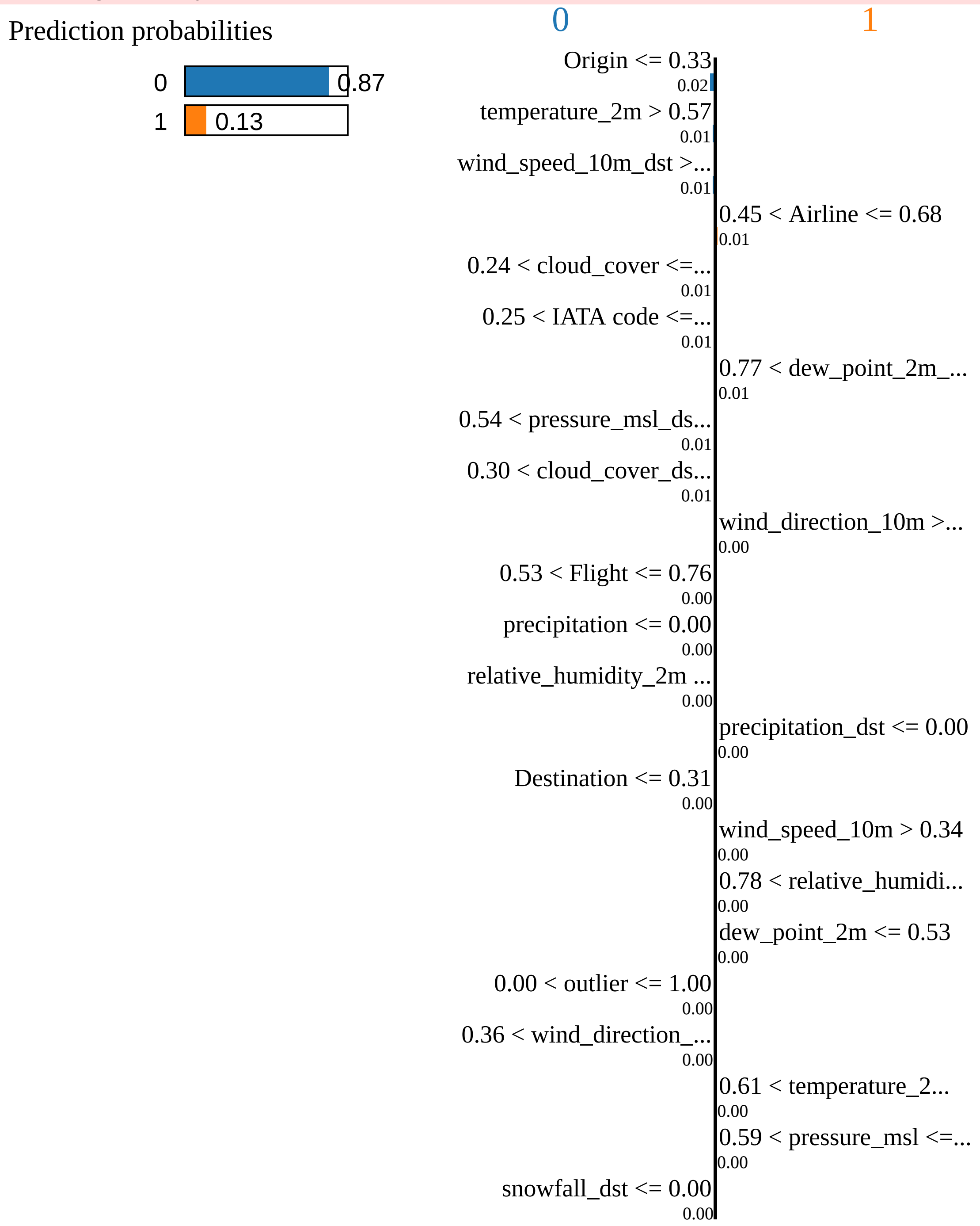


| Feature | Value |
|---|---|
| Origin | 0.78 |
| temperature_2m | 0.59 |
| relative_humidity_2m_dst | 0.43 |
| precipitation | 0.00 |
| wind_direction_10m_dst | 0.28 |
| cloud_cover | 0.00 |
| dew_point_2m | 0.55 |
| Airline | 0.31 |
| Destination | 0.13 |

```
Índice: 684
Intercept 0.11771234273502038
```

```
Prediction_local [0.06764218]
Right: 0.1287254246245056
```

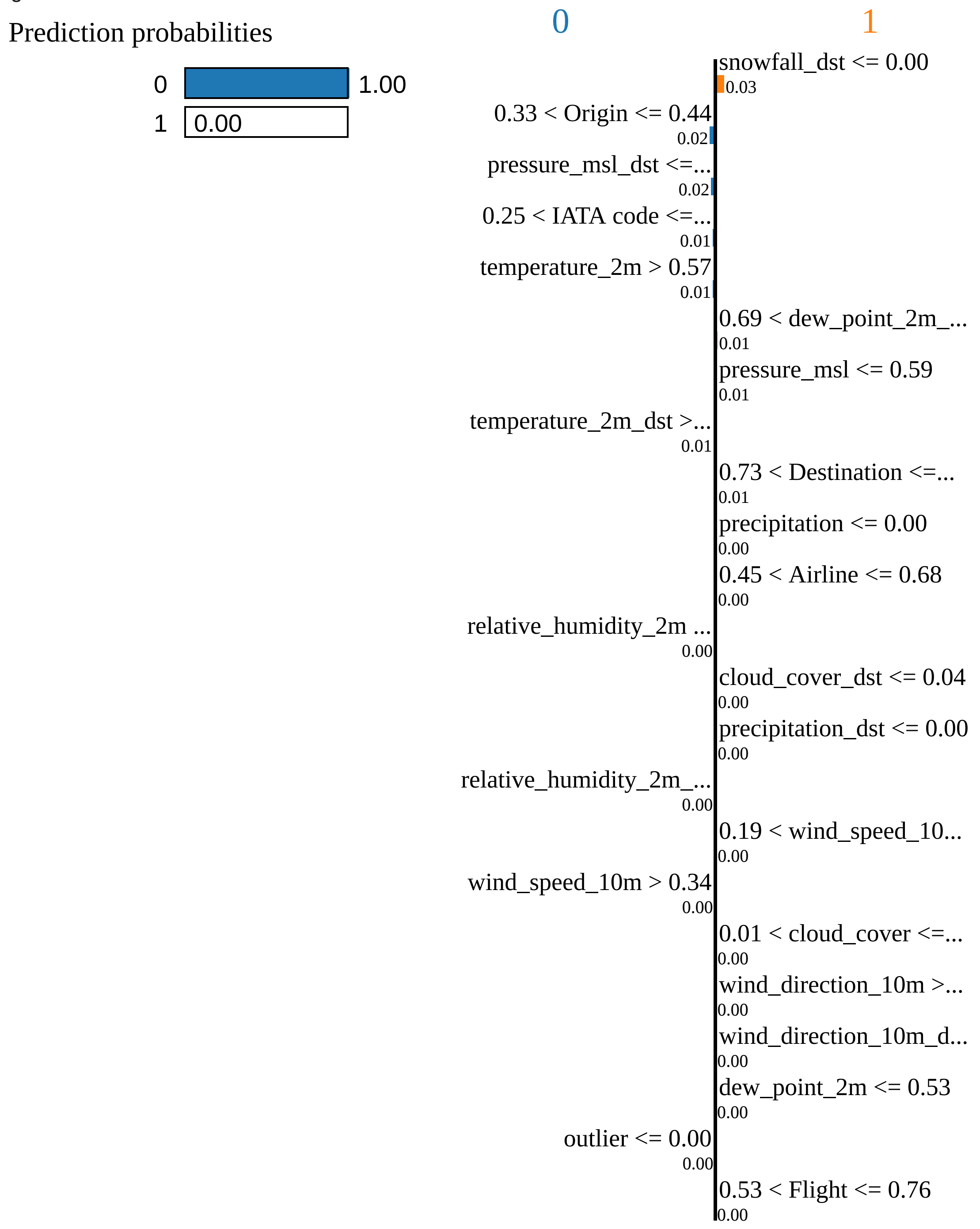
```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```



| Feature | Value |
|---|---|
| Origin | 0.11 |
| temperature_2m | 0.65 |
| wind_speed_10m_dst | 0.58 |
| Airline | 0.65 |
| cloud_cover | 0.31 |
| IATA code | 0.32 |
| dew_point_2m_dst | 0.84 |
| pressure_msl_dst | 0.58 |

|  |  |
|---|---|
| cloud_cover_dst | 0.31 |

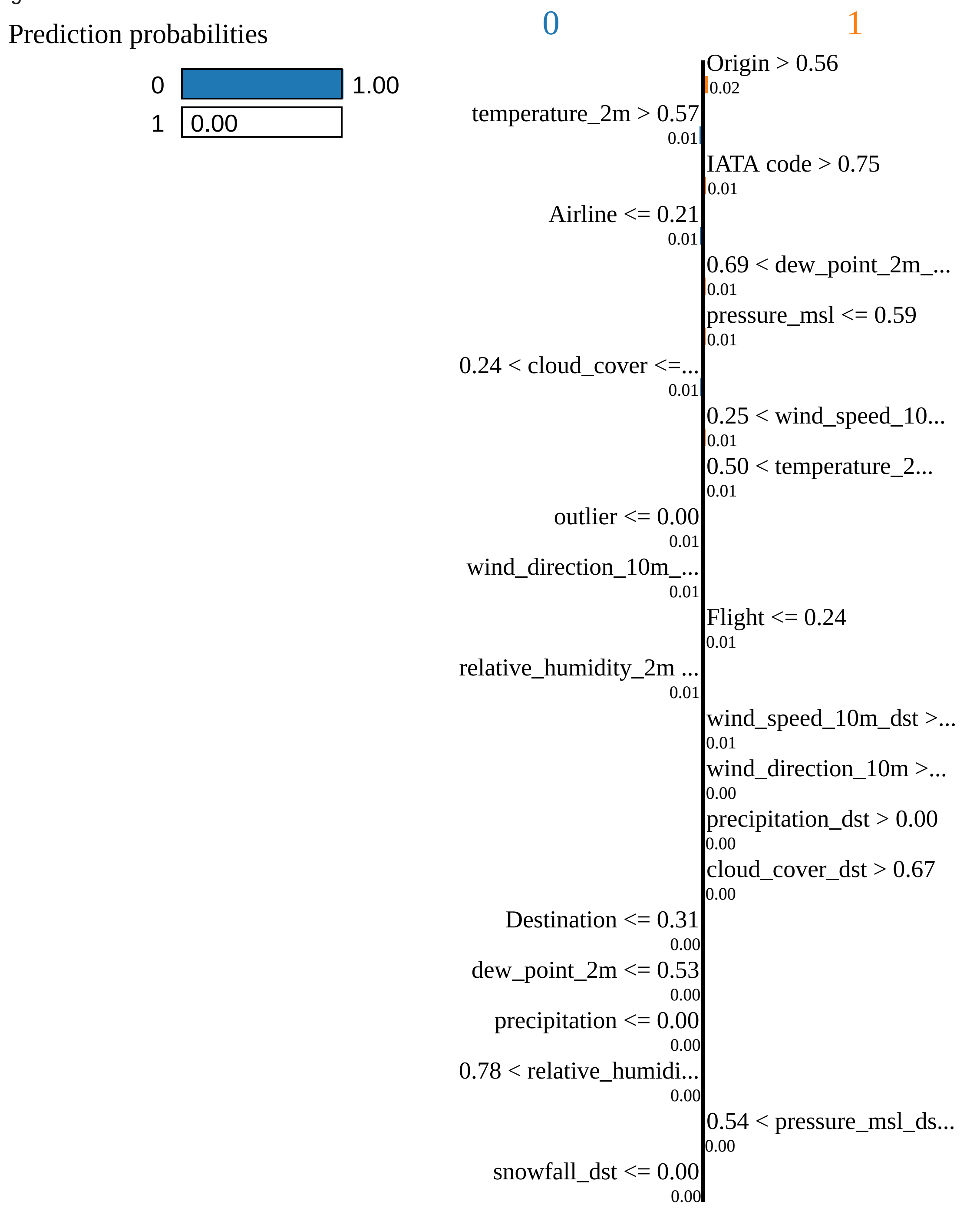
```
Índice: 4684
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
Intercept 0.07136345568901041
Prediction_local [0.07655881]
Right: 0.0
```



| Feature | Value |
|---|---|
| snowfall_dst | 0.00 |
| Origin | 0.44 |
| pressure_msl_dst | 0.46 |
| IATA code | 0.35 |
| temperature_2m | 0.73 |
| dew_point_2m_dst | 0.72 |

|  |  |
|---|---|
| pressure_msl | 0.44 |
| temperature_2m_dst | 0.95 |
| Destination | 0.79 |

```
Índice: 6234
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
Intercept 0.09582034177025894
Prediction_local [0.12585688]
Right: 0.0
```

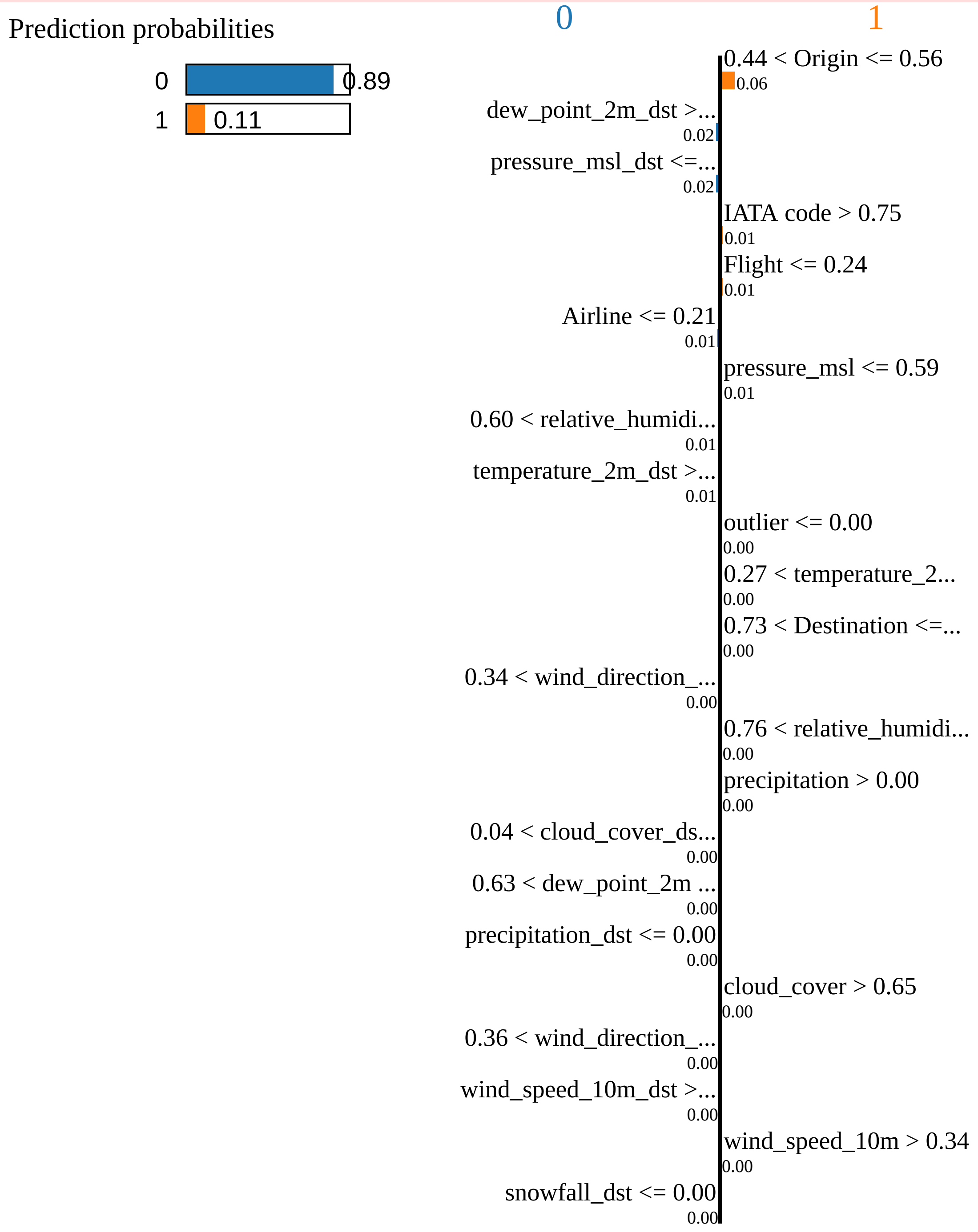


| Feature | Value |
|---|---|
| Origin | 1.00 |
| temperature_2m | 0.77 |
| IATA code | 0.98 |
| Airline | 0.20 |
| dew_point_2m_dst | 0.76 |

| | |
|---|---|
| pressure_msl | 0.59 |
| cloud_cover | 0.55 |
| wind_speed_10m | 0.30 |
| temperature_2m_dst | 0.50 |
| outlier | 0.00 |

```
Índice: 3652
Intercept 0.1002333426939606
Prediction_local [0.14440626]
Right: 0.11088888888888888
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

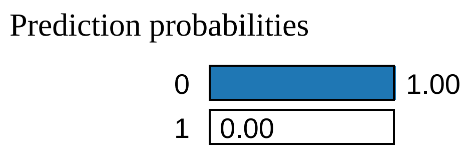


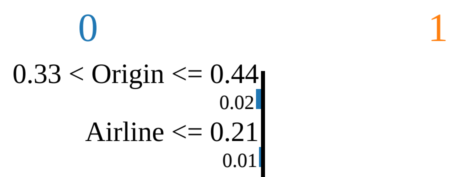
| Feature | Value |
|---|---|
| Origin | 0.56 |
| dew_point_2m_dst | 0.90 |
| pressure_msl_dst | 0.53 |

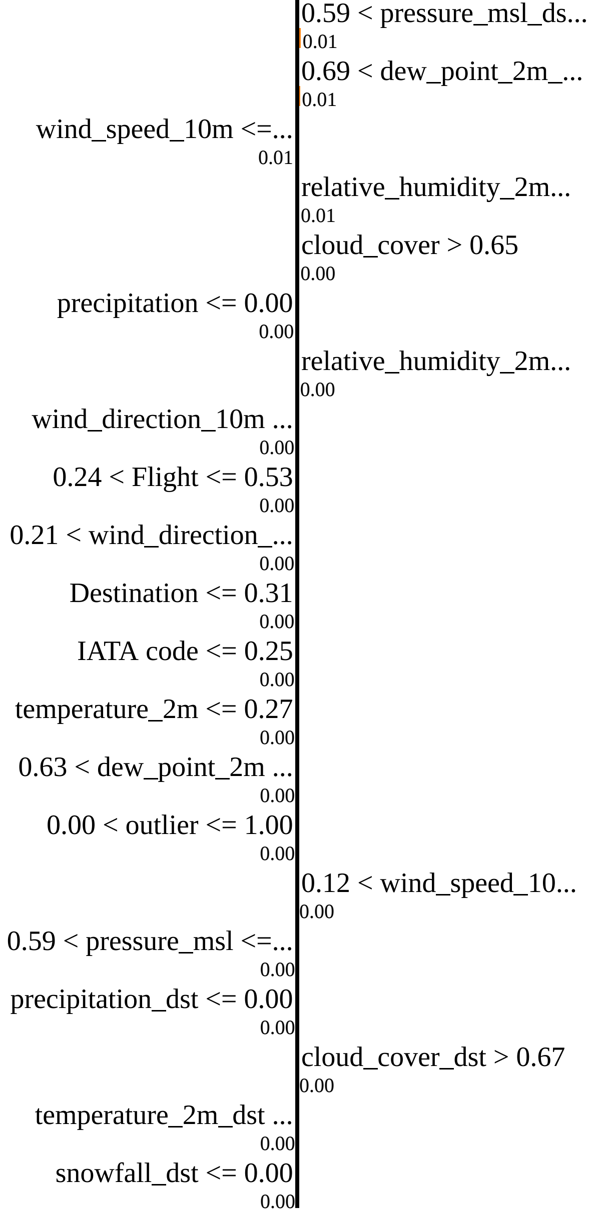
| | |
|---|---|
| IATA code | 0.80 |
| Flight | 0.06 |
| Airline | 0.21 |
| pressure_msl | 0.38 |
| relative_humidity_2m_dst | 0.74 |
| temperature_2m_dst | 0.71 |
| outlier | 0.00 |

```
Índice: 5779
Intercept 0.11286988570070013
Prediction_local [0.08624492]
Right: 0.0
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

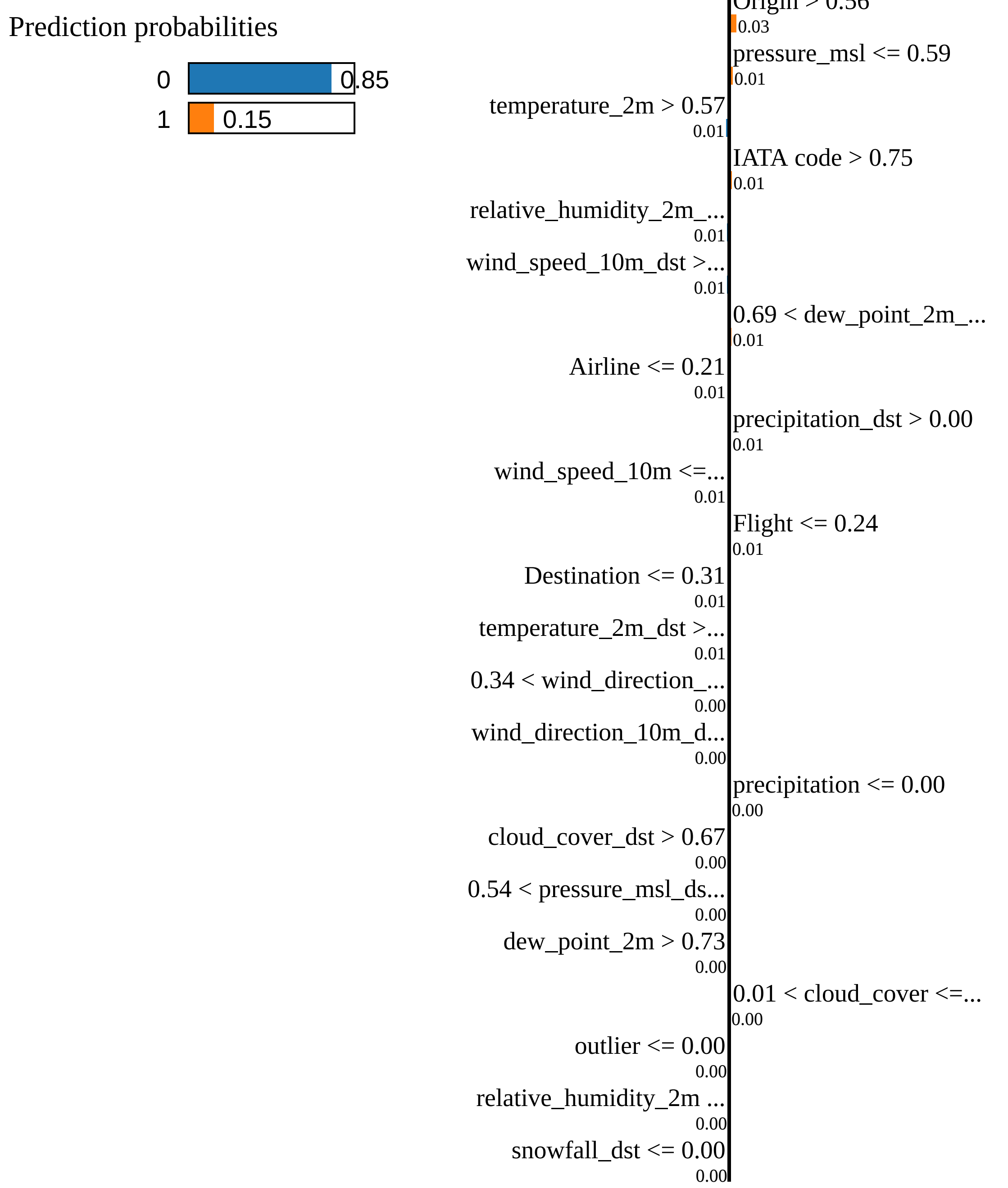






| Feature | Value |
|---|---|
| Origin | 0.44 |
| Airline | 0.03 |
| pressure_msl_dst | 0.63 |
| dew_point_2m_dst | 0.74 |
| wind_speed_10m | 0.17 |
| relative_humidity_2m | 1.00 |
| cloud_cover | 1.00 |
| precipitation | 0.00 |
| relative_humidity_2m_dst | 0.98 |
| wind_direction_10m | 0.18 |

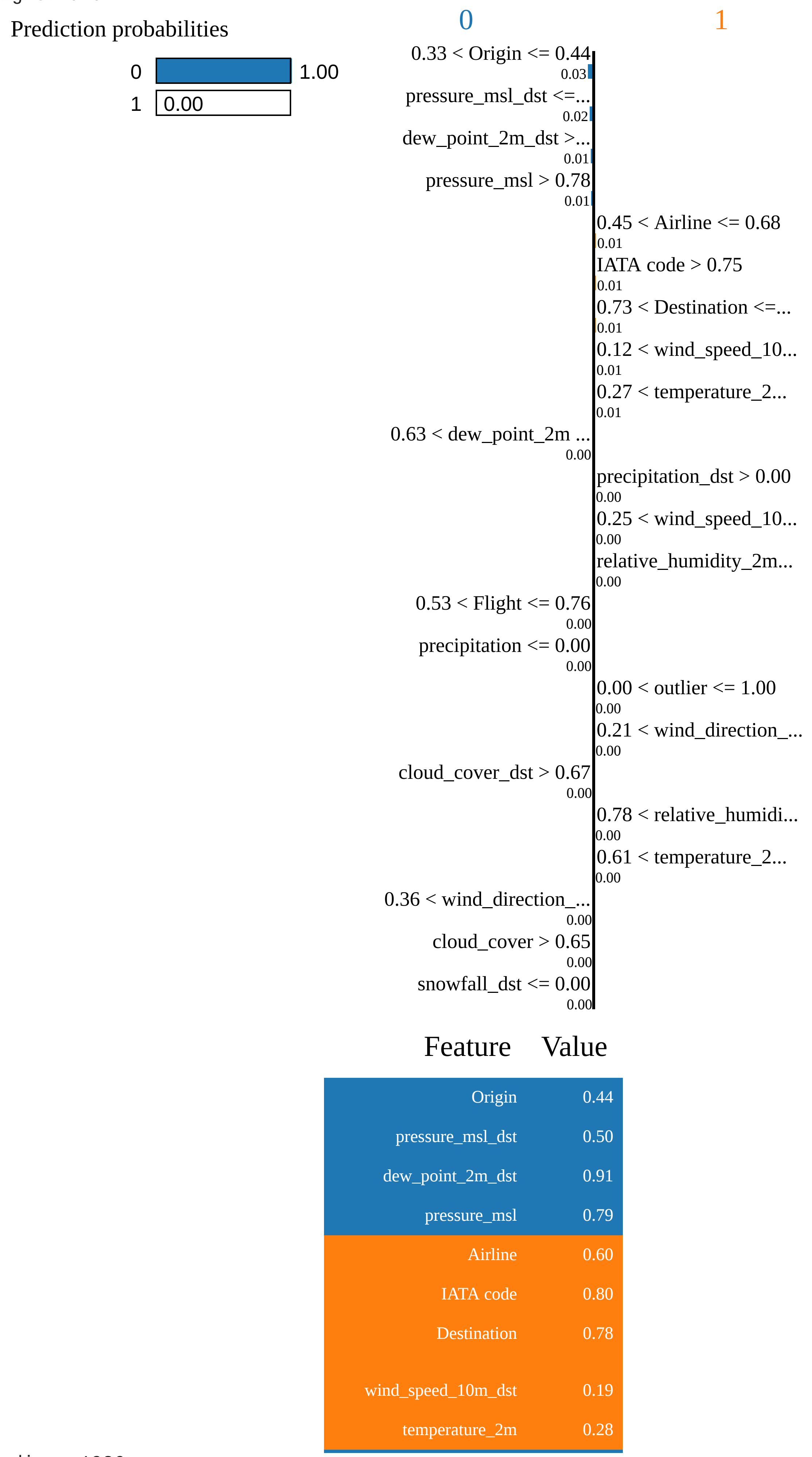
```
Índice: 5068
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
Intercept 0.10149994182703106
Prediction_local [0.10694654]
Right: 0.1496415770609319
```



| Feature | Value |
|---|---|
| Origin | 0.78 |
| pressure_msl | 0.53 |
| temperature_2m | 0.85 |
| IATA code | 0.96 |
| relative_humidity_2m_dst | 0.48 |
| wind_speed_10m_dst | 0.37 |
| dew_point_2m_dst | 0.77 |
| Airline | 0.20 |
| precipitation_dst | 0.01 |
| wind_speed_10m | 0.05 |

```
Índice: 4239
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
Intercept 0.11354019606563744
```

```
Prediction_local [0.08693064]
Right: 0.0
```

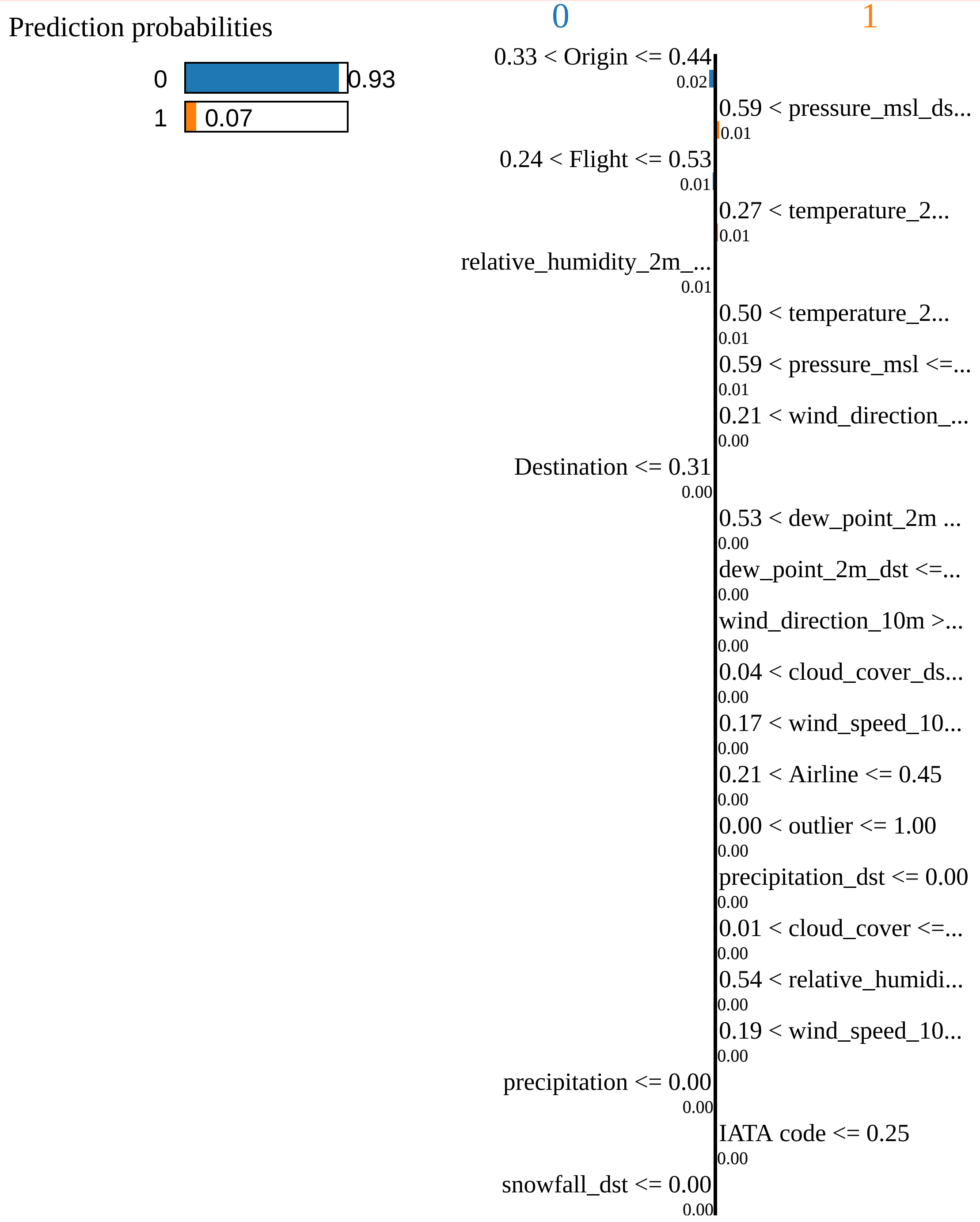


| Feature | Value |
|---|---|
| Origin | 0.44 |
| pressure_msl_dst | 0.50 |
| dew_point_2m_dst | 0.91 |
| pressure_msl | 0.79 |
| Airline | 0.60 |
| IATA code | 0.80 |
| Destination | 0.78 |
| wind_speed_10m_dst | 0.19 |
| temperature_2m | 0.28 |

```
Índice: 1986
Intercept 0.10009223549495952
```

```
Prediction_local [0.11642707]
Right: 0.06599023099900533
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

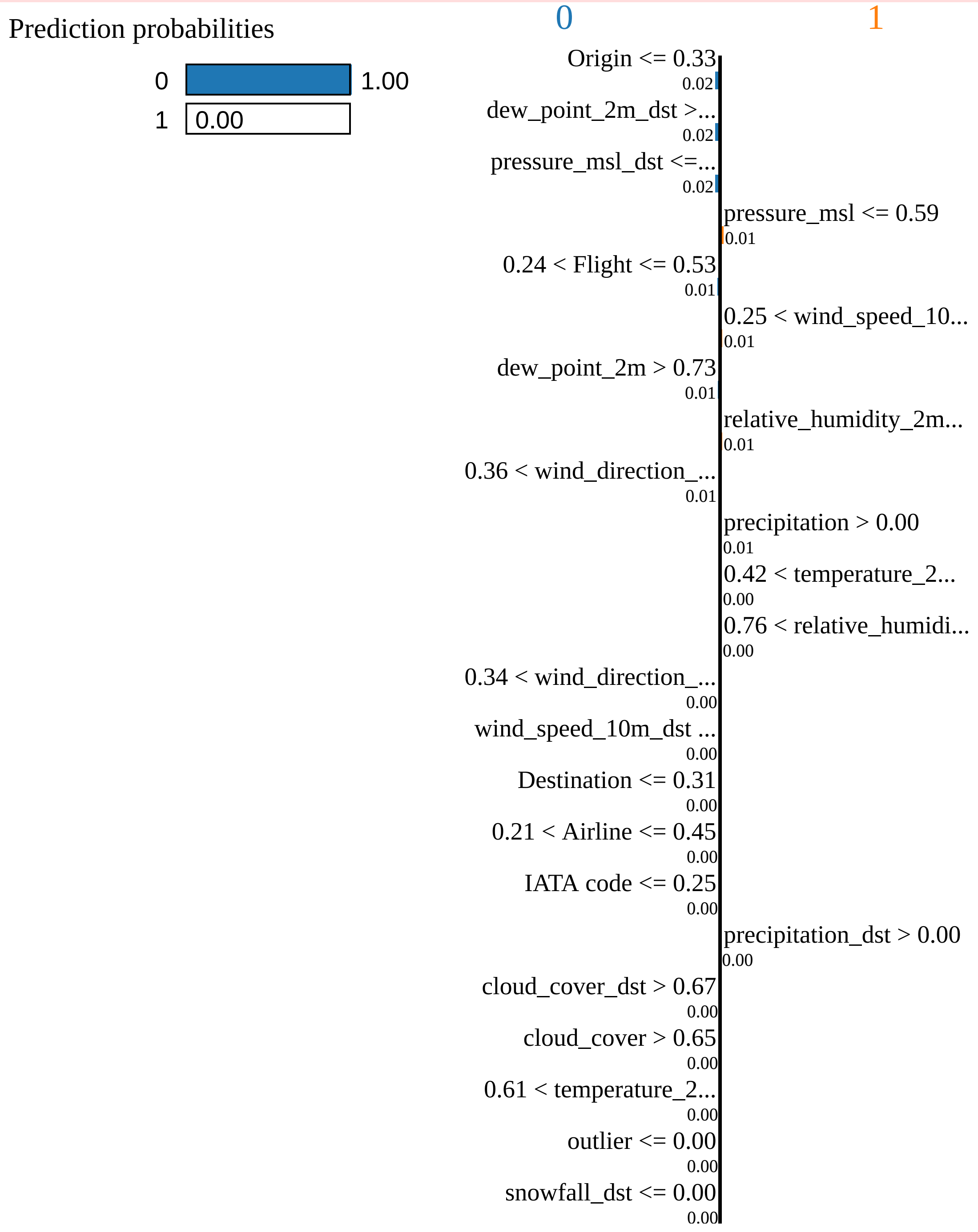


| Feature | Value |
|---|---|
| Origin | 0.44 |
| pressure_msl_dst | 0.60 |
| Flight | 0.47 |
| temperature_2m | 0.38 |
| relative_humidity_2m_dst | 0.57 |
| temperature_2m_dst | 0.58 |
| pressure_msl | 0.65 |
| wind_direction_10m_dst | 0.27 |

| | |
|---|---|
| Destination | 0.10 |

```
Índice: 10828
Intercept 0.11665953687129374
Prediction_local [0.06726476]
Right: 0.0
```

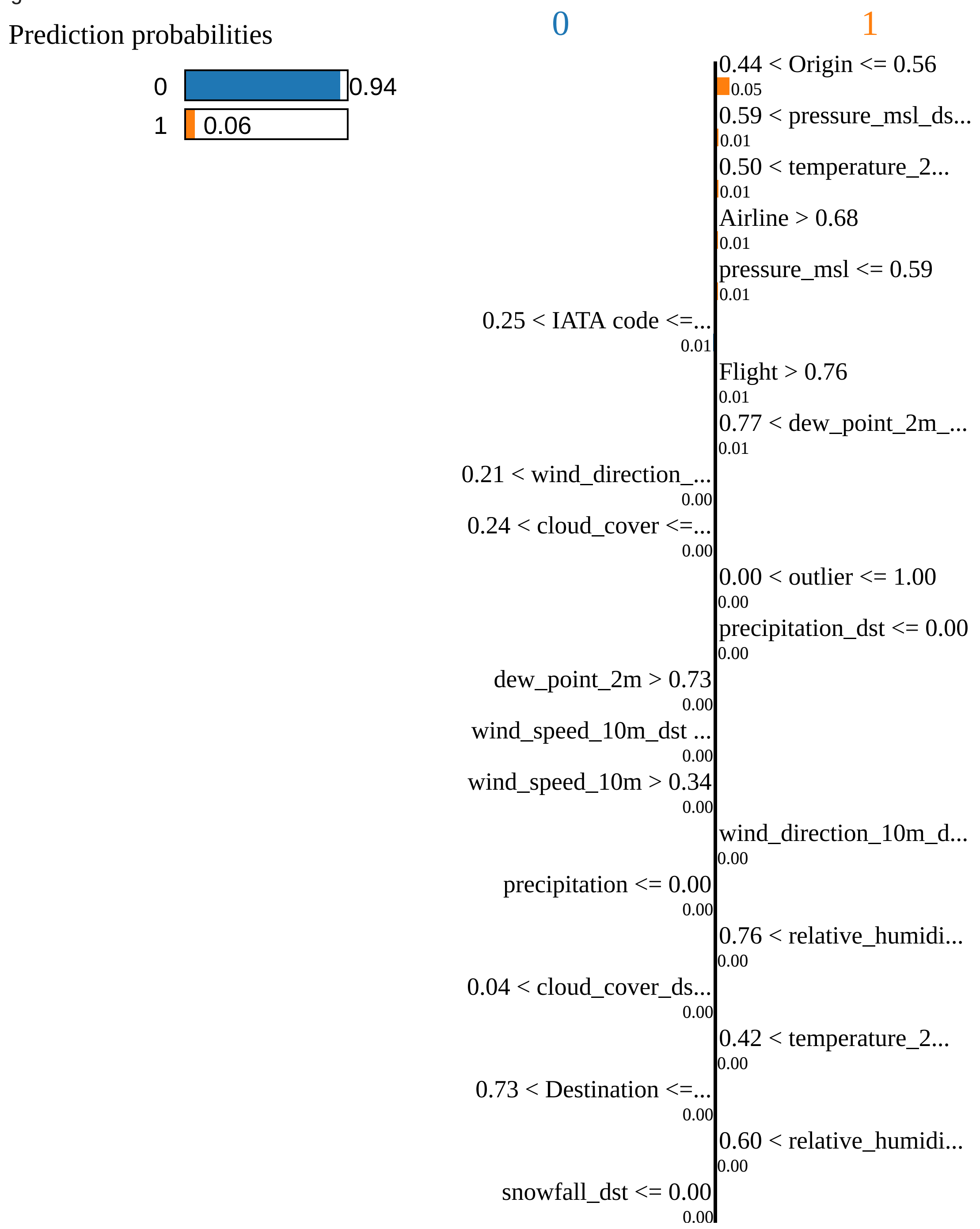
```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```



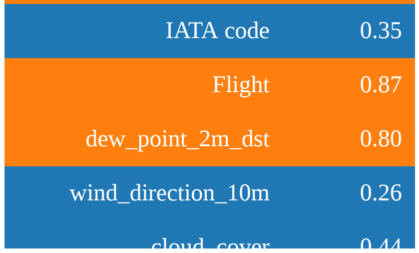
| Feature | Value |
|---|---|
| Origin | 0.33 |
| dew_point_2m_dst | 0.91 |
| pressure_msl_dst | 0.51 |
| pressure_msl | 0.57 |
| Flight | 0.35 |
| wind_speed_10m | 0.30 |

| | |
|---|---|
| dew_point_2m | 0.87 |
| relative_humidity_2m_dst | 0.92 |
| wind_direction_10m_dst | 0.49 |

```
Índice: 856
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
Intercept 0.08220402183879667
Prediction_local [0.17667996]
Right: 0.05761251741860697
```



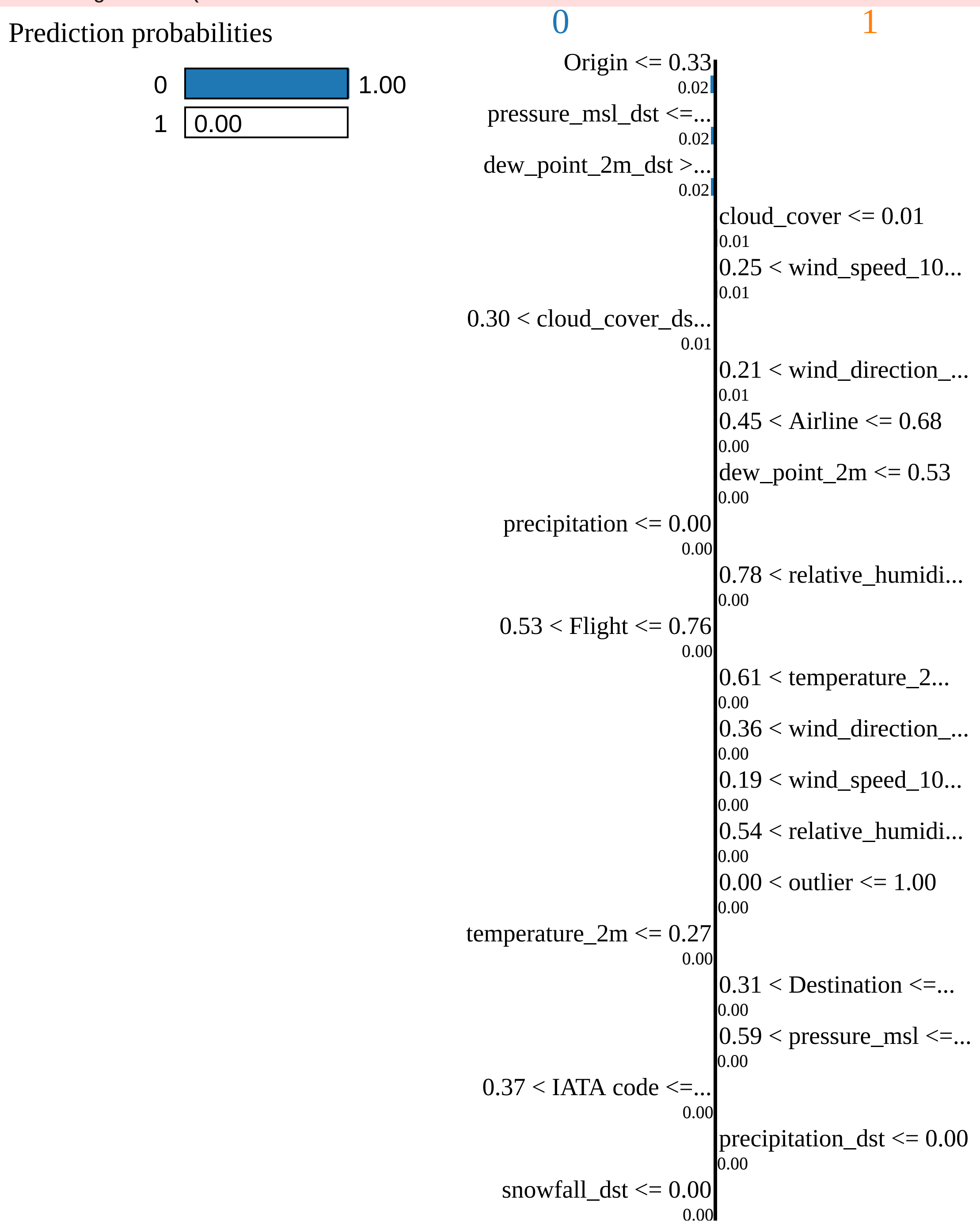
| Feature | Value |
|---|---|
| Origin | 0.56 |
| pressure_msl_dst | 0.63 |
| temperature_2m_dst | 0.60 |
| Airline | 0.89 |
| pressure_msl | 0.54 |

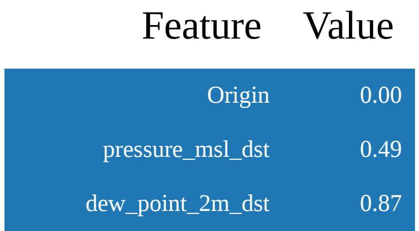


| | |
|---|---|
| IATA code | 0.35 |
| Flight | 0.87 |
| dew_point_2m_dst | 0.80 |
| wind_direction_10m | 0.26 |
| cloud_cover | 0.44 |

```
Índice: 11883
Intercept 0.10852768047332971
Prediction_local [0.09112968]
Right: 0.0005229250334672021
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

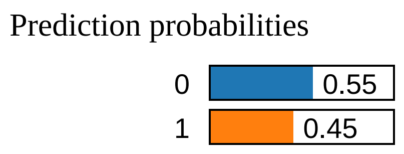


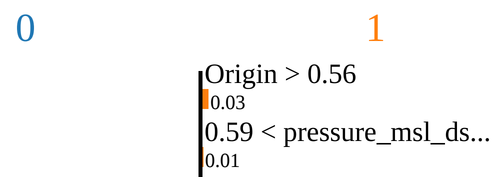


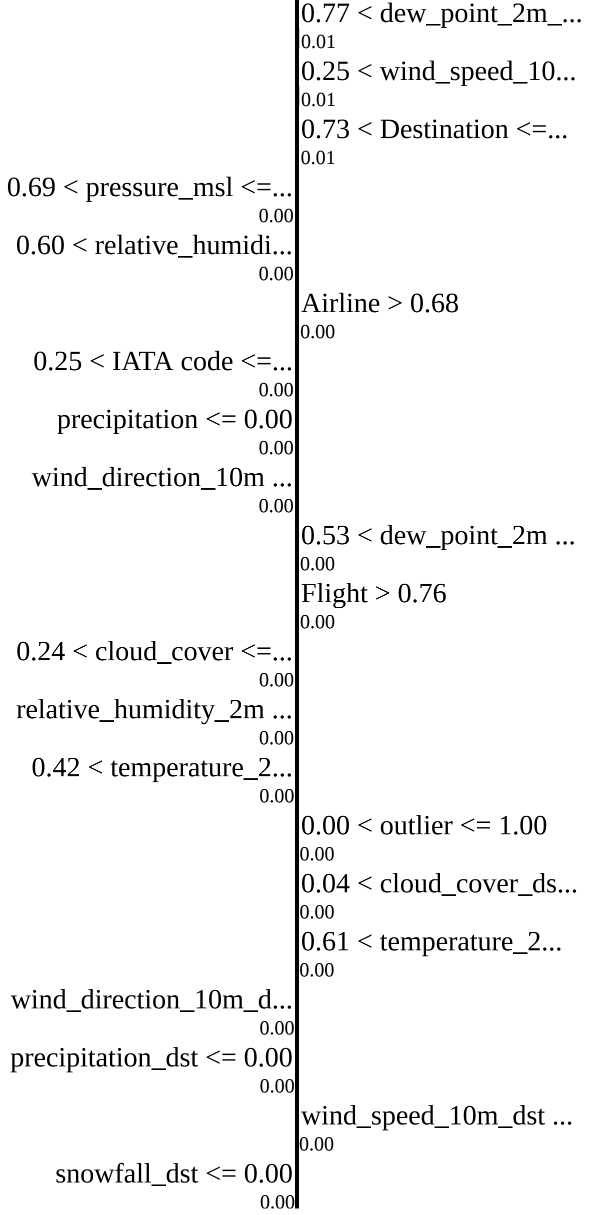
| Feature | Value |
|---|---|
| Origin | 0.00 |
| pressure_msl_dst | 0.49 |
| dew_point_2m_dst | 0.87 |

| | |
|---|---|
| cloud_cover | 0.00 |
| wind_speed_10m | 0.33 |
| cloud_cover_dst | 0.35 |
| wind_direction_10m | 0.26 |
| Airline | 0.51 |
| dew_point_2m | 0.43 |
| precipitation | 0.00 |

```
Índice: 9357
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
Intercept 0.09702414663183226
Prediction_local [0.13845863]
Right: 0.44777373723896474
```



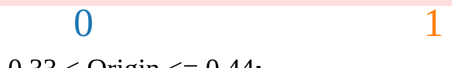


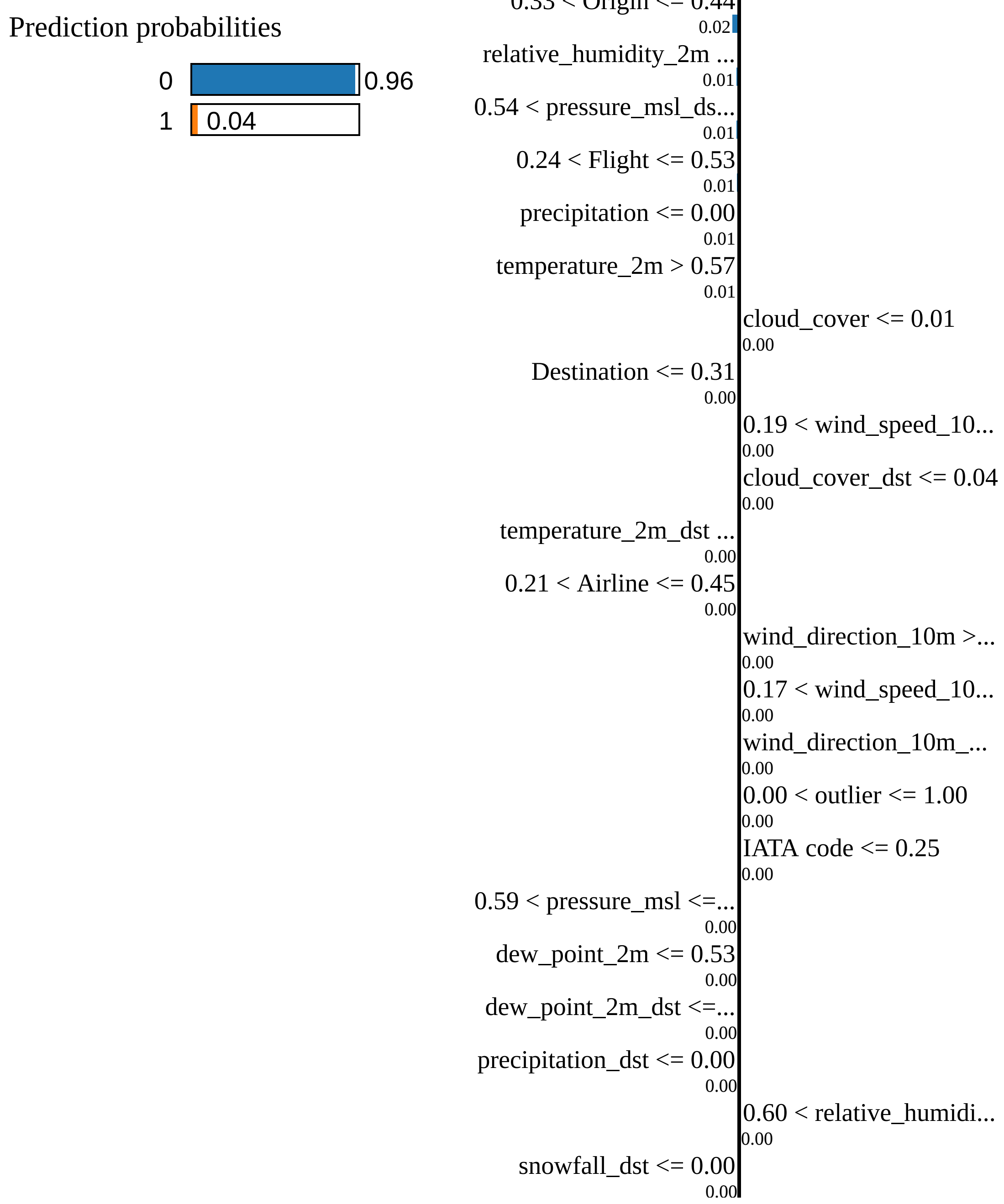


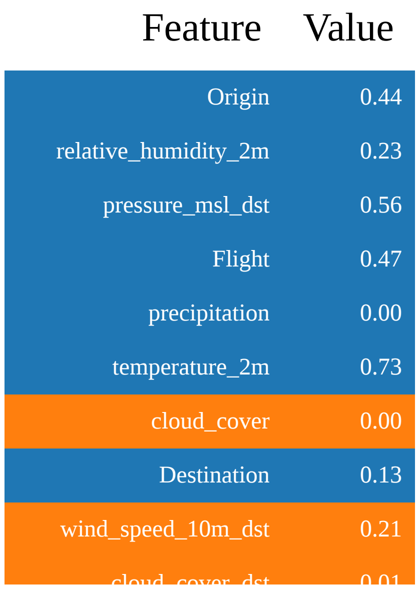
| Feature | Value |
|---|---|
| Origin | 1.00 |
| pressure_msl_dst | 0.61 |
| dew_point_2m_dst | 0.83 |
| wind_speed_10m | 0.29 |
| Destination | 0.79 |
| pressure_msl | 0.71 |
| relative_humidity_2m_dst | 0.71 |
| Airline | 0.97 |
| IATA code | 0.35 |
| precipitation | 0.00 |

```
Índice: 12206
Intercept 0.119972481696423
Prediction_local [0.06701651]
Right: 0.0365665896843726
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```



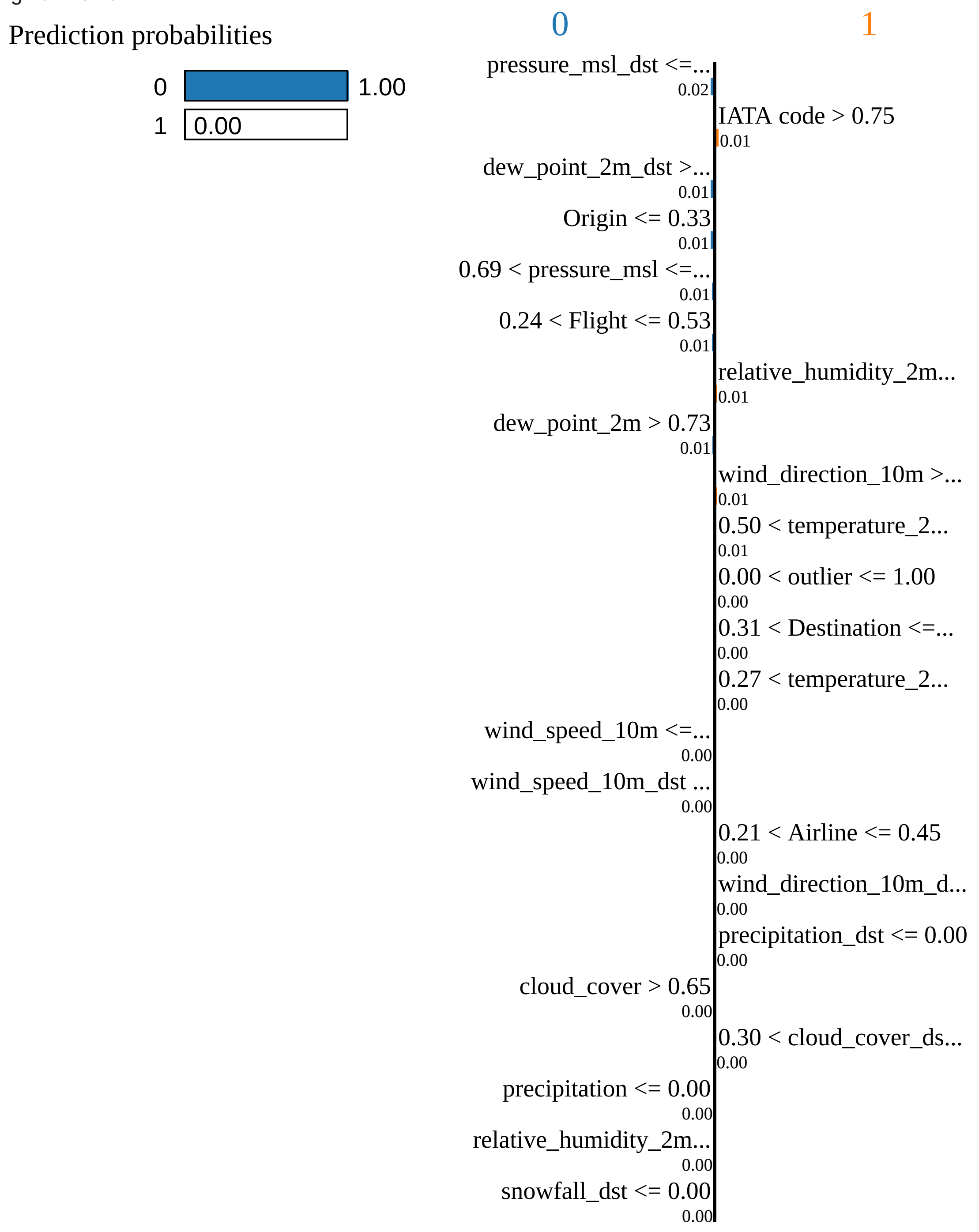




| Feature | Value |
|---|---|
| Origin | 0.44 |
| relative_humidity_2m | 0.23 |
| pressure_msl_dst | 0.56 |
| Flight | 0.47 |
| precipitation | 0.00 |
| temperature_2m | 0.73 |
| cloud_cover | 0.00 |
| Destination | 0.13 |
| wind_speed_10m_dst | 0.21 |
| cloud_cover_dst | 0.01 |

```
Índice: 10711
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
Intercept 0.10601091892017815
```

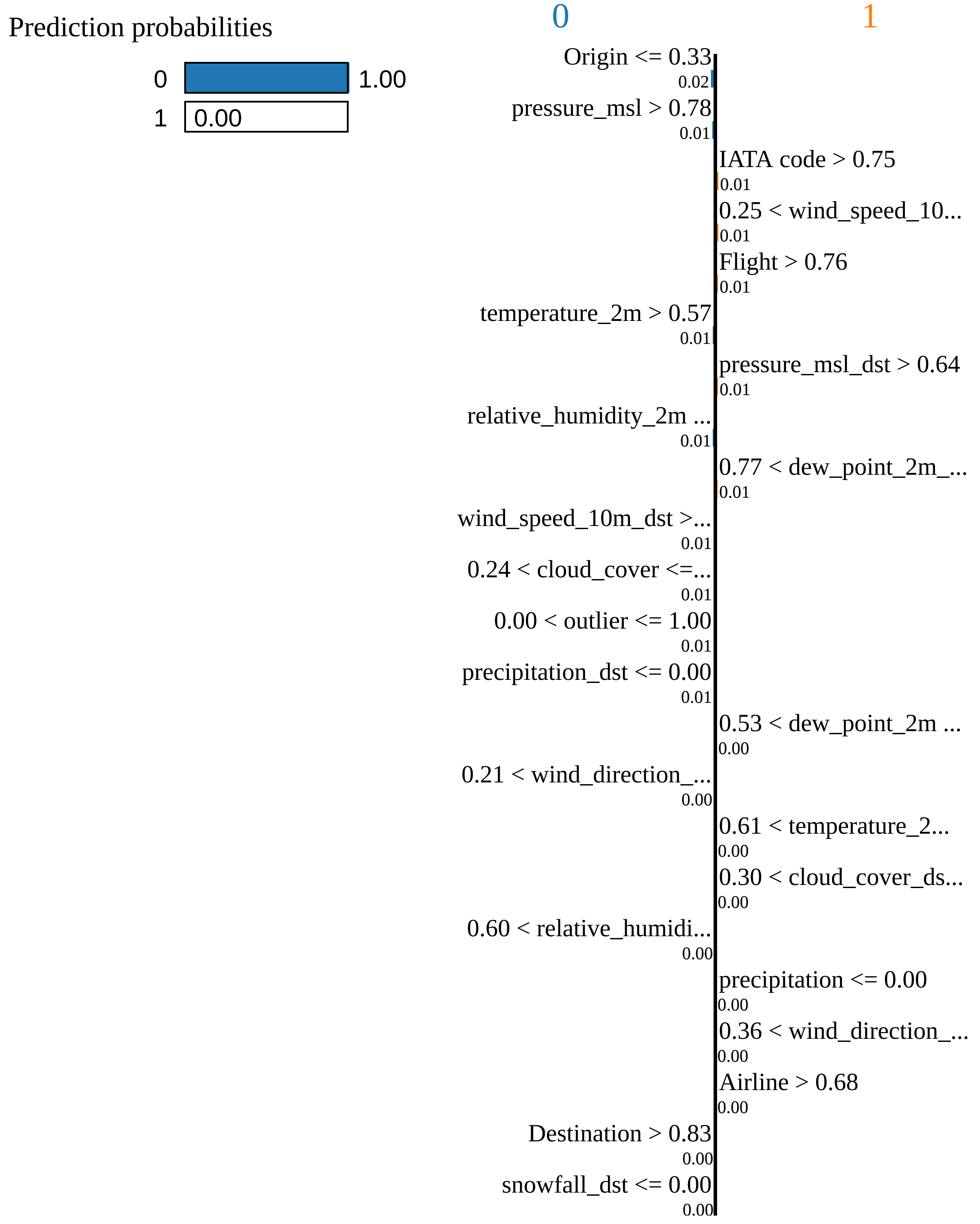
```
Prediction_local [0.08245142]
Right: 0.0
```



| Feature | Value |
|---|---|
| pressure_msl_dst | 0.46 |
| IATA code | 0.76 |
| dew_point_2m_dst | 0.87 |
| Origin | 0.33 |
| pressure_msl | 0.71 |
| Flight | 0.49 |
| relative_humidity_2m | 0.93 |
| dew_point_2m | 0.82 |
| wind_direction_10m | 0.82 |

```
Índice: 8226
```

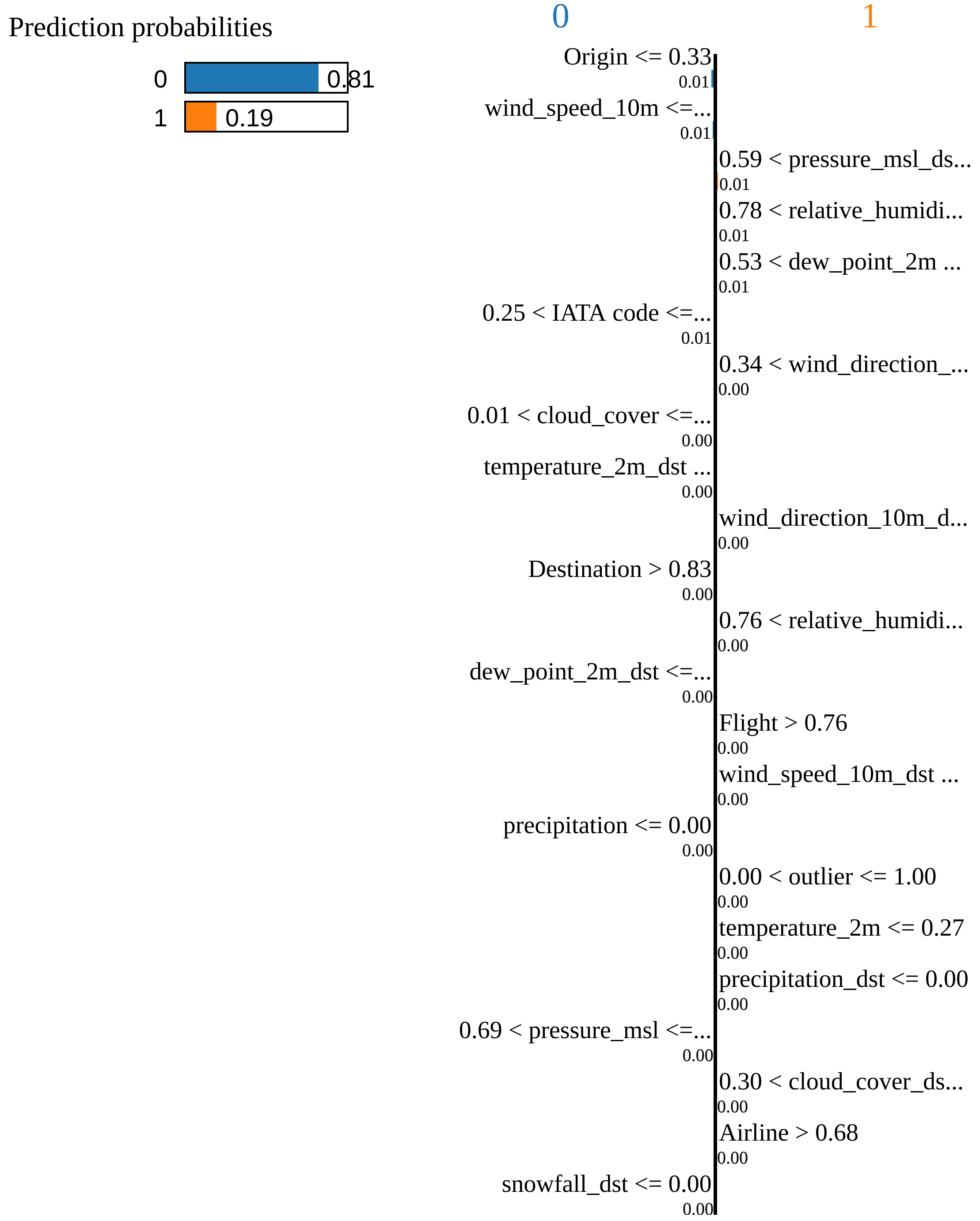
```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
Intercept 0.11230663613466377
Prediction_local [0.09891606]
Right: 0.0
```



| Feature | Value |
|---|---|
| Origin | 0.33 |
| pressure_msl | 0.83 |
| IATA code | 0.97 |
| wind_speed_10m | 0.32 |
| Flight | 0.87 |
| temperature_2m | 0.61 |
| pressure_msl_dst | 0.68 |
| relative_humidity_2m | 0.45 |

|  |  |
|---|---|
| dew_point_2m_dst | 0.80 |
| wind_speed_10m_dst | 0.48 |

```
Índice: 14168
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
Intercept 0.10200418983226547
Prediction_local [0.09707387]
Right: 0.19083639055777754
```

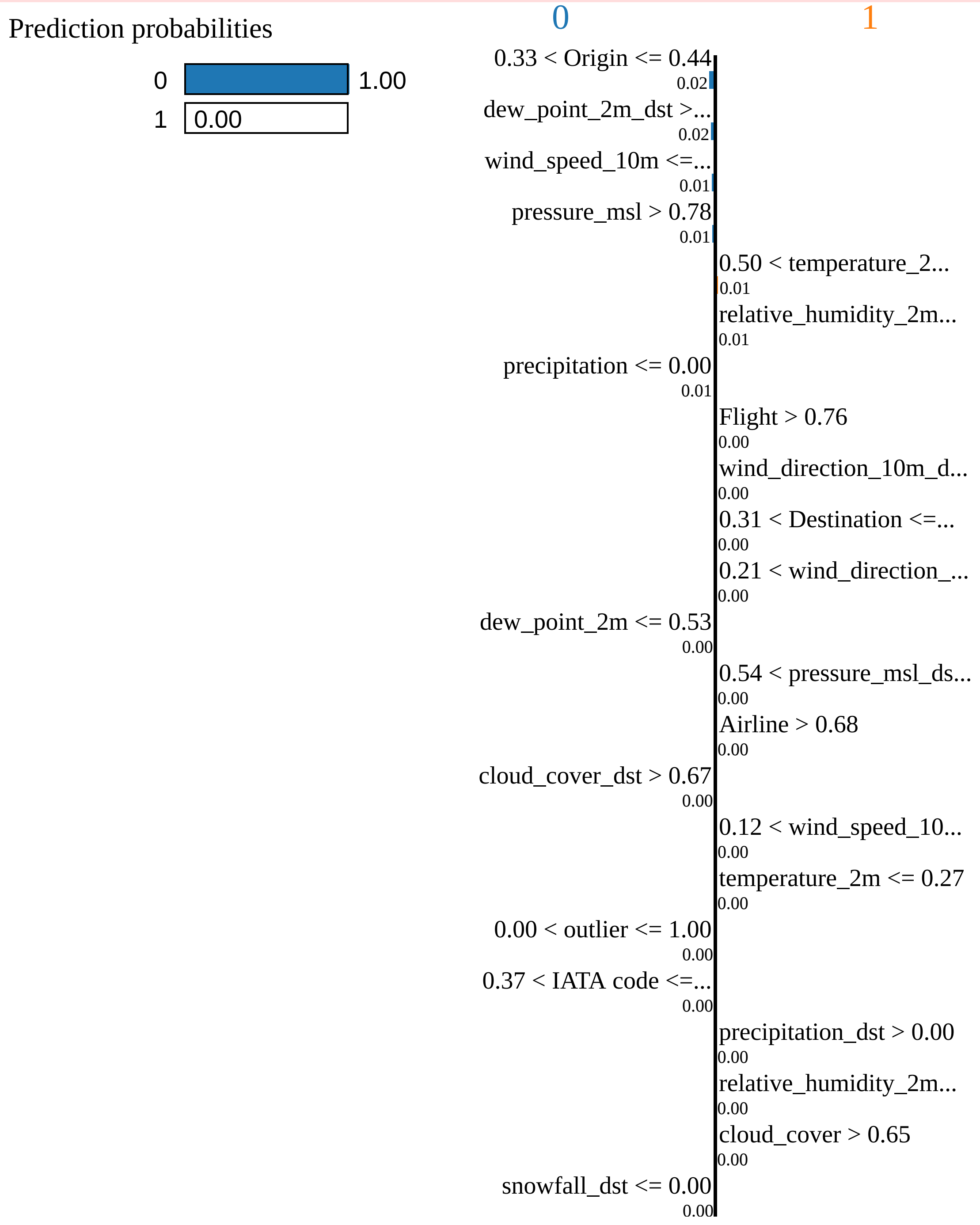


| Feature | Value |
|---|---|
| Origin | 0.22 |
| wind_speed_10m | 0.13 |
| pressure_msl_dst | 0.61 |
| relative_humidity_2m_dst | 0.79 |
| dew_point_2m | 0.54 |
| IATA code | 0.37 |

| | |
|---|---|
| wind_direction_10m | 0.54 |
| cloud_cover | 0.22 |
| temperature_2m_dst | 0.36 |

```
Índice: 2945
Intercept 0.11744527820171931
Prediction_local [0.08072308]
Right: 0.0
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

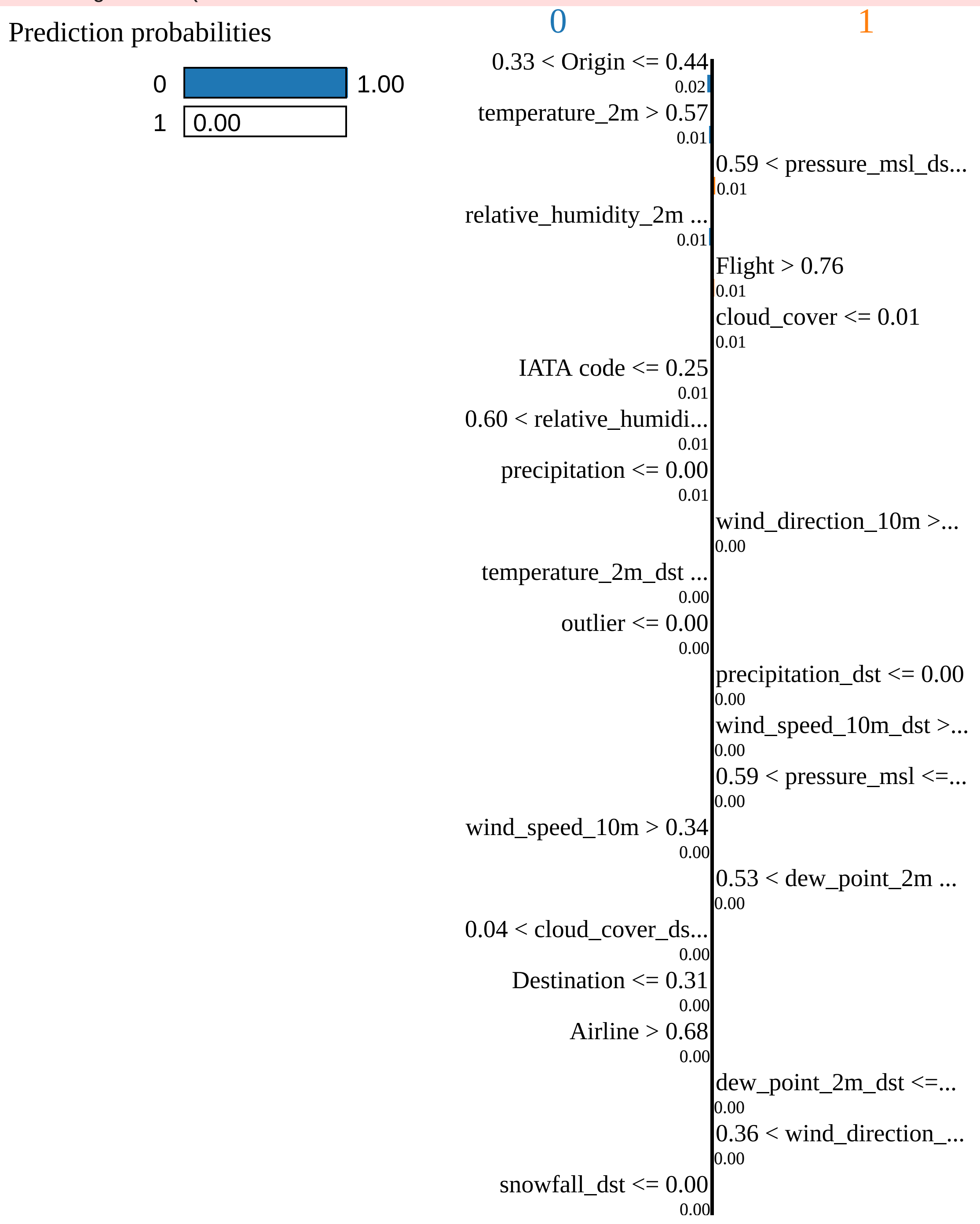


| Feature | Value |
|---|---|
| Origin | 0.44 |
| dew_point_2m_dst | 0.88 |
| wind_speed_10m | 0.13 |
| pressure_msl | 0.88 |

| | |
|---|---|
| temperature_2m_dst | 0.60 |
| relative_humidity_2m | 0.98 |
| precipitation | 0.00 |
| Flight | 0.88 |
| wind_direction_10m_dst | 0.71 |

```
Índice: 11188
Intercept 0.11361592369793228
Prediction_local [0.08306711]
Right: 0.0
```

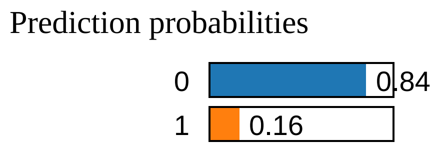
```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

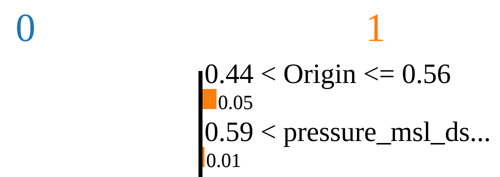


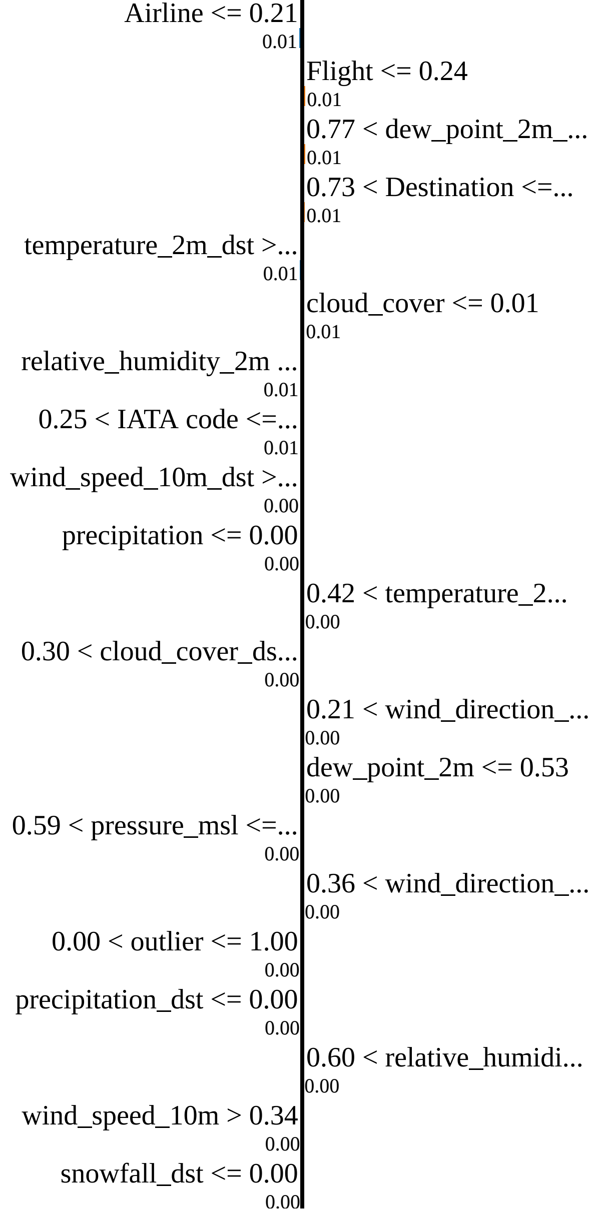
| Feature | Value |
|---|---|
| Origin | 0.44 |
| temperature_2m | 0.60 |

| | |
|---|---|
| pressure_msl_dst | 0.62 |
| relative_humidity_2m | 0.42 |
| Flight | 0.85 |
| cloud_cover | 0.00 |
| IATA code | 0.01 |
| relative_humidity_2m_dst | 0.62 |
| precipitation | 0.00 |

```
Índice: 13337
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
Intercept 0.095854326972034
Prediction_local [0.16203337]
Right: 0.15884222737062995
```

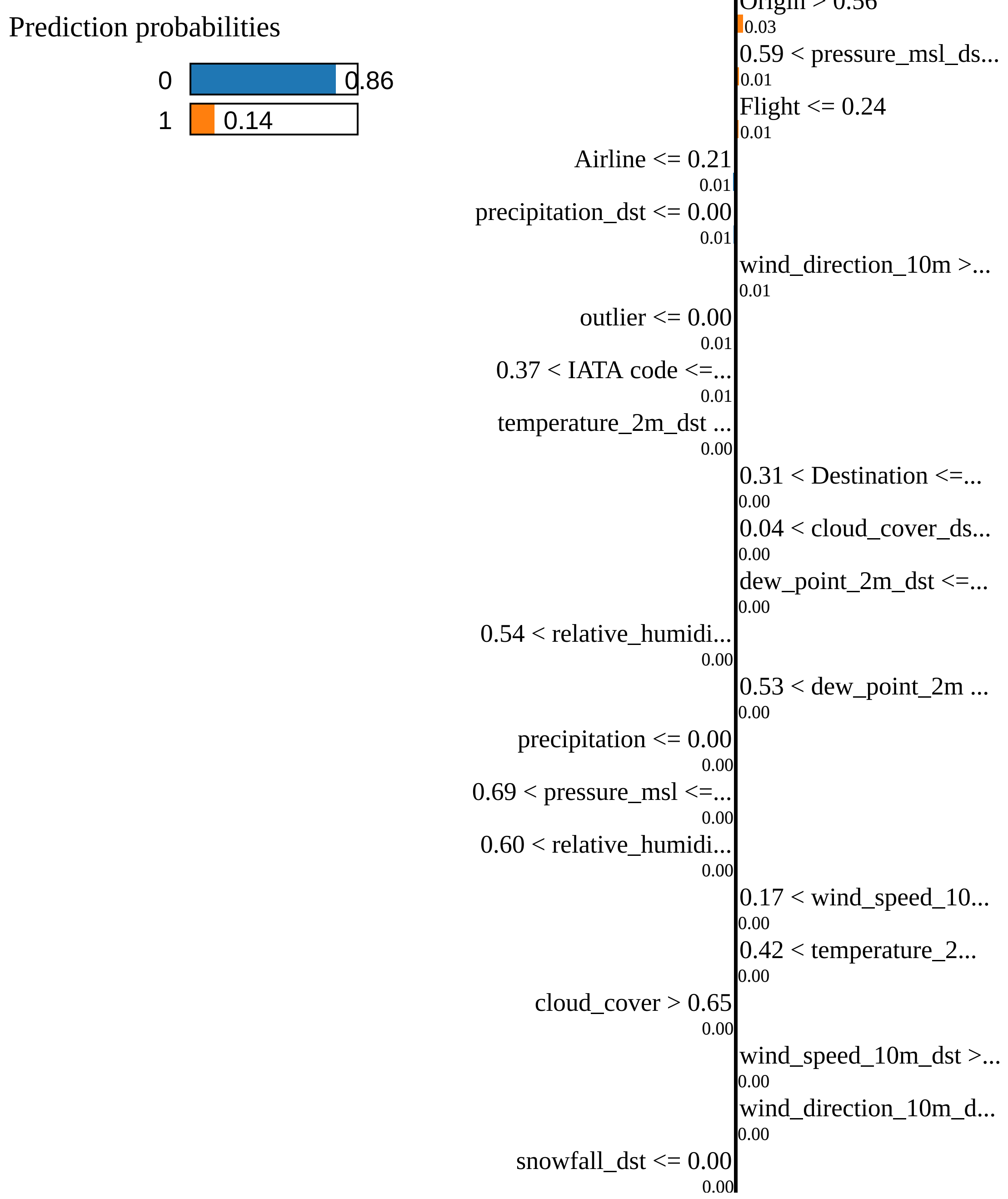






| Feature | Value |
|---|---|
| Origin | 0.56 |
| pressure_msl_dst | 0.61 |
| Airline | 0.12 |
| Flight | 0.23 |
| dew_point_2m_dst | 0.85 |
| Destination | 0.79 |
| temperature_2m_dst | 0.72 |
| cloud_cover | 0.00 |
| relative_humidity_2m | 0.51 |
| IATA code | 0.35 |

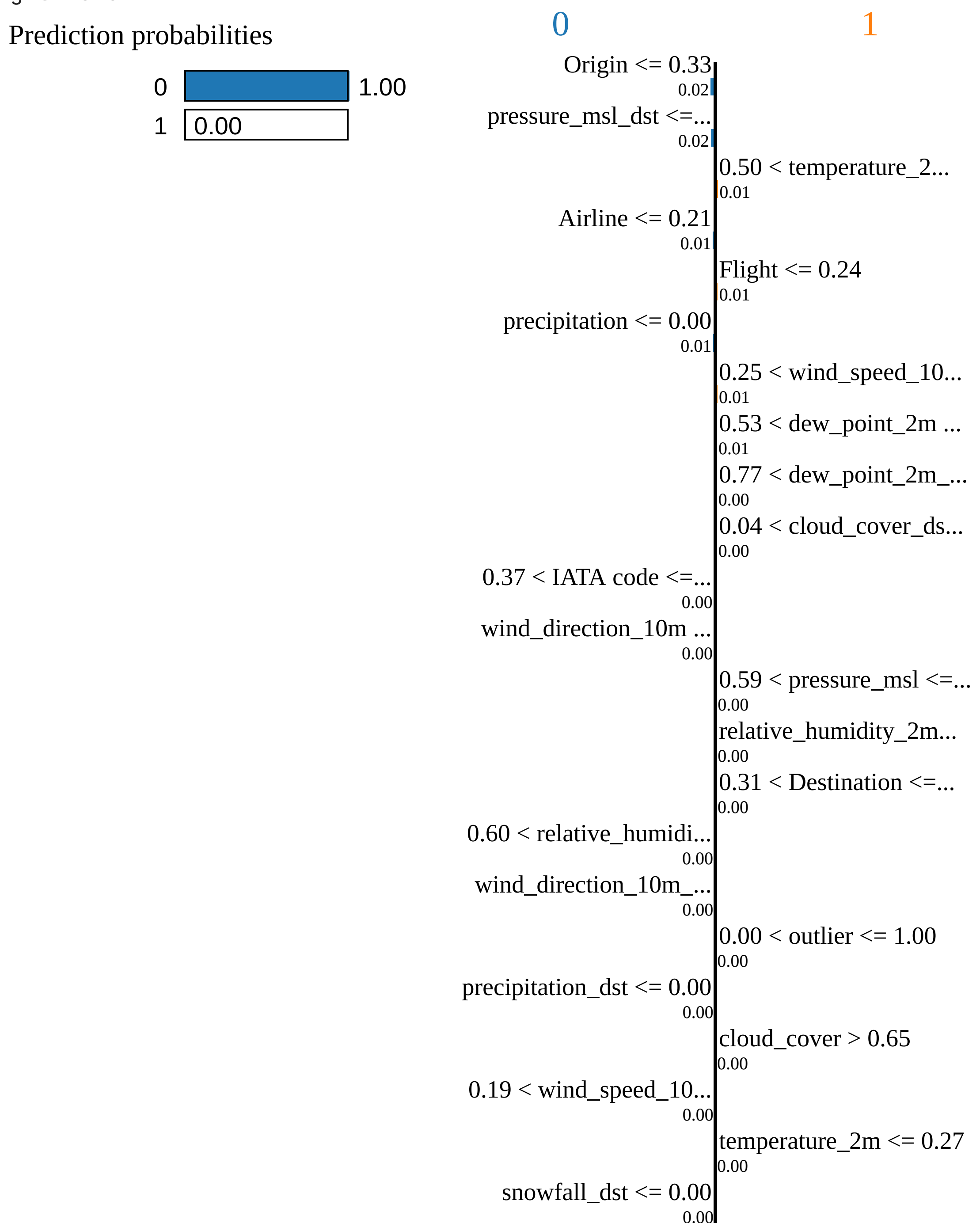
```
Índice: 14218
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
Intercept 0.10159135749179135
Prediction_local [0.12913211]
Right: 0.142135023029933
```

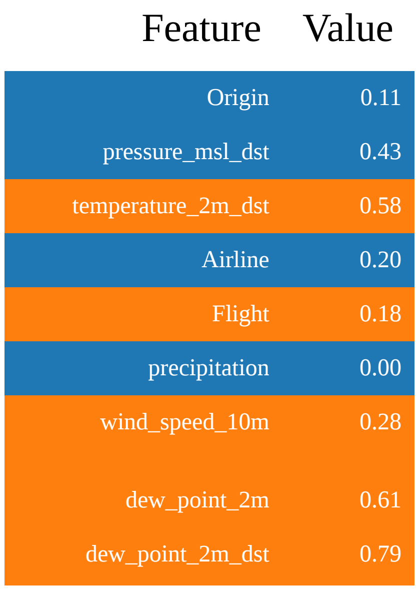


| Feature | Value |
|---|---|
| Origin | 1.00 |
| pressure_msl_dst | 0.63 |
| Flight | 0.16 |
| Airline | 0.20 |
| precipitation_dst | 0.00 |
| wind_direction_10m | 0.96 |
| outlier | 0.00 |
| IATA code | 0.75 |
| temperature_2m_dst | 0.46 |
| Destination | 0.73 |

```
Índice: 5969
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
Intercept 0.11056987028148901
```

```
Prediction_local [0.09841408]
Right: 0.0
```

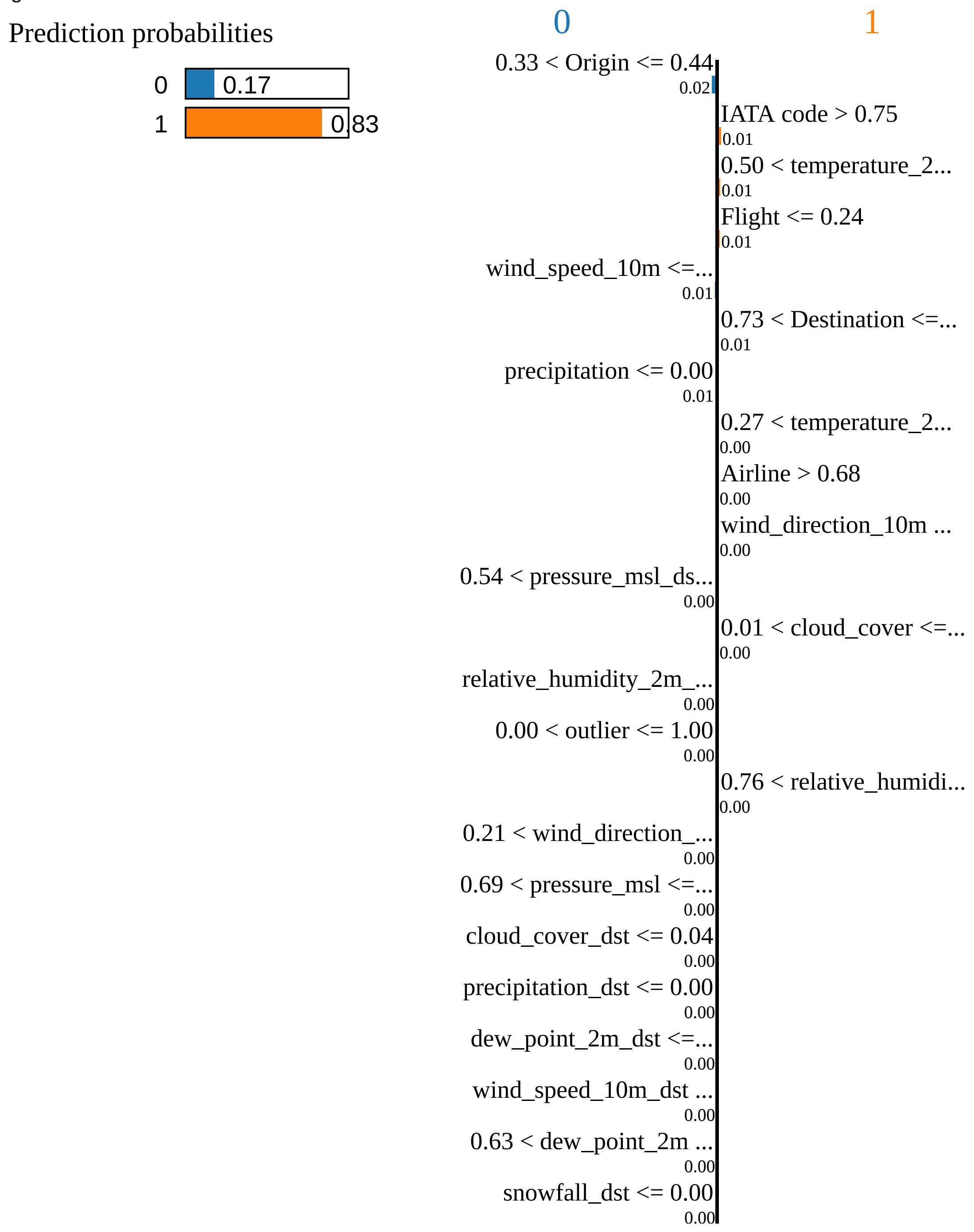




| Feature | Value |
|---|---|
| Origin | 0.11 |
| pressure_msl_dst | 0.43 |
| temperature_2m_dst | 0.58 |
| Airline | 0.20 |
| Flight | 0.18 |
| precipitation | 0.00 |
| wind_speed_10m | 0.28 |
| dew_point_2m | 0.61 |
| dew_point_2m_dst | 0.79 |

```
Índice: 9082
```

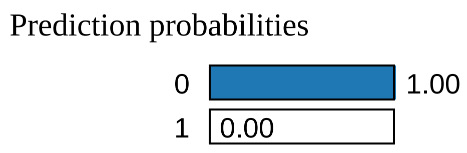
```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
Intercept 0.10773102521307218
Prediction_local [0.11459072]
Right: 0.827611786152953
```

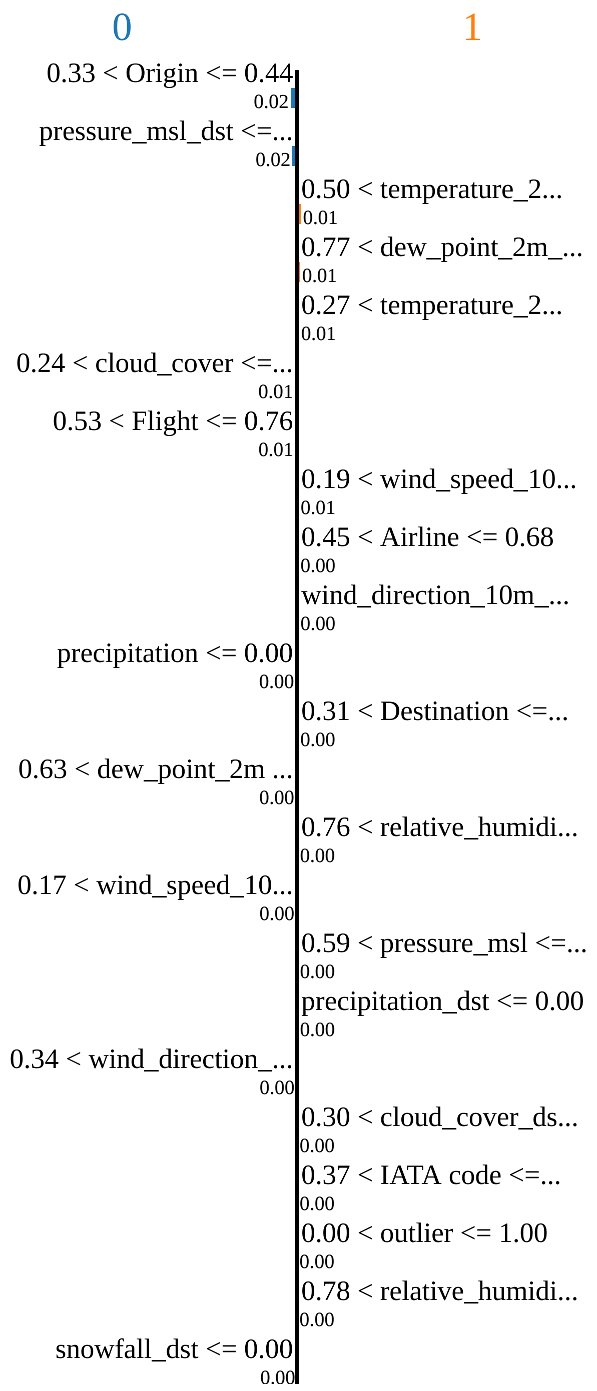


| Feature | Value |
|---|---|
| Origin | 0.44 |
| IATA code | 0.78 |
| temperature_2m_dst | 0.60 |
| Flight | 0.00 |
| wind_speed_10m | 0.15 |
| Destination | 0.78 |
| precipitation | 0.00 |
| temperature_2m | 0.29 |

|  |  |
|---|---|
| Airline | 0.99 |

```
Índice: 10121
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
Intercept 0.103624713814972 79
Prediction_local [0.10134633]
Right: 0.0
```

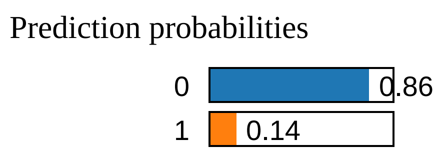


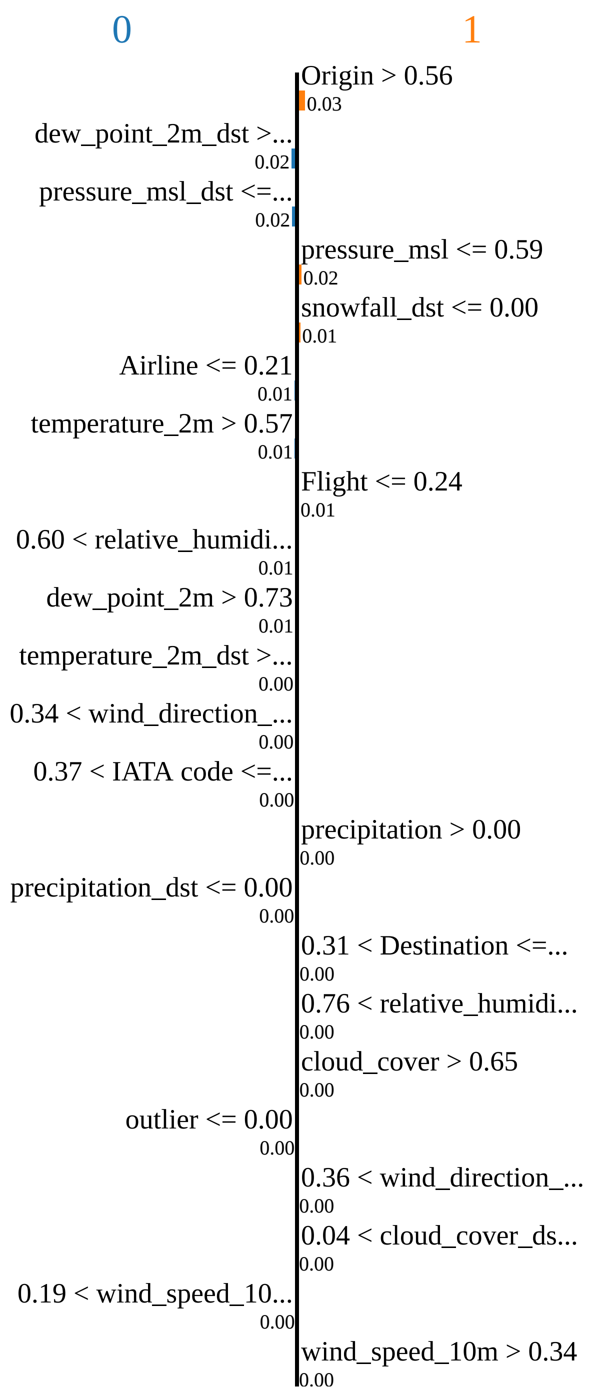


| Feature | Value |
|---|---|
| Origin | 0.44 |
| pressure_msl_dst | 0.47 |
| temperature_2m_dst | 0.57 |
| dew_point_2m_dst | 0.83 |
| temperature_2m | 0.39 |
| cloud_cover | 0.59 |

| | |
|---|---|
| Flight | 0.72 |
| wind_speed_10m_dst | 0.22 |
| Airline | 0.65 |

```
Índice: 7528
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
Intercept 0.09733508609941549
Prediction_local [0.08902051]
Right: 0.14254979770940193
```

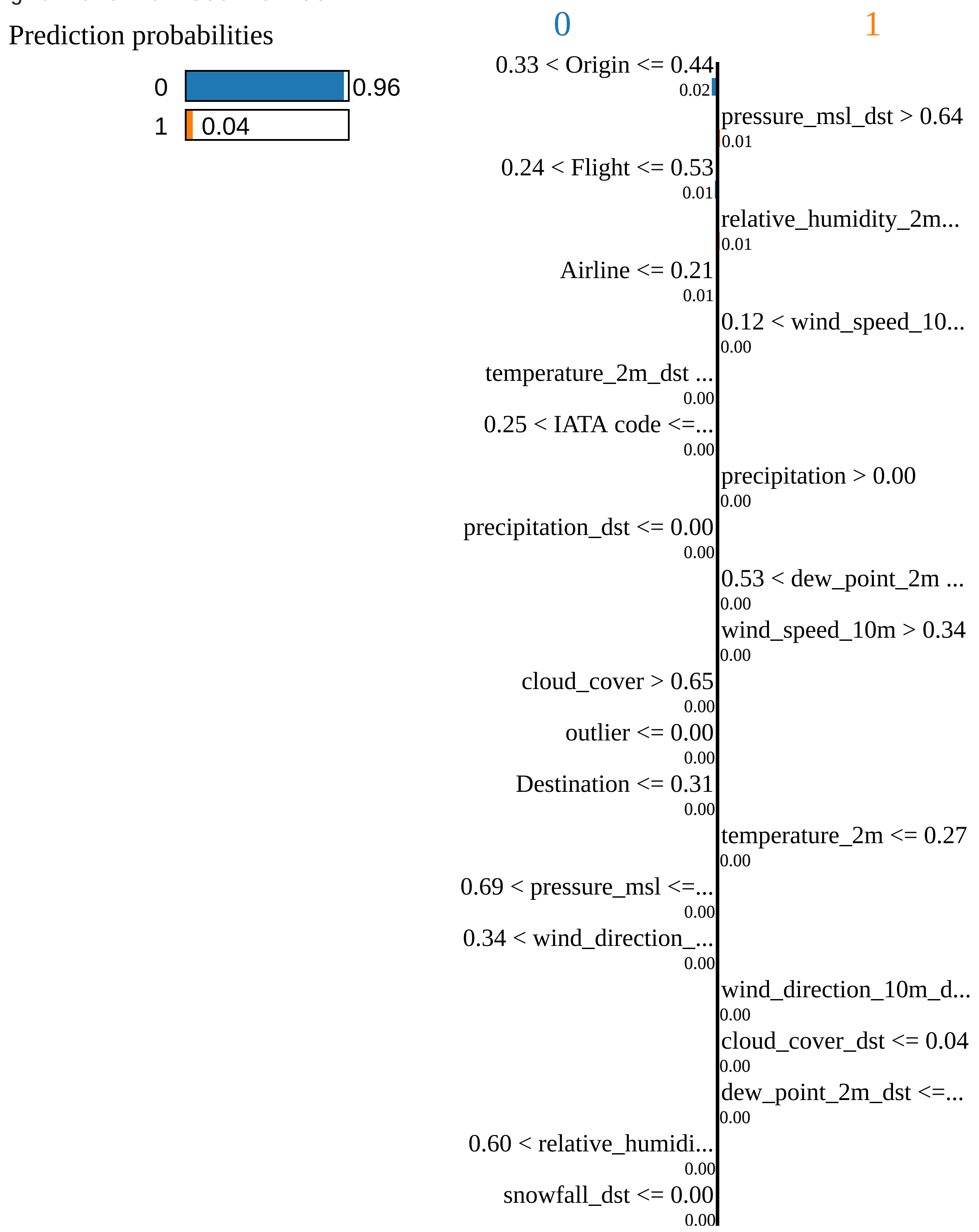




| Feature | Value |
|---|---|
| Origin | 0.67 |
| dew_point_2m_dst | 0.89 |
| pressure_msl_dst | 0.52 |
| pressure_msl | 0.54 |

| | |
|---|---|
| snowfall_dst | 0.00 |
| Airline | 0.20 |
| temperature_2m | 0.58 |
| Flight | 0.08 |
| relative_humidity_2m_dst | 0.63 |

```
Índice: 4026
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
Intercept 0.11394410461564468
Prediction_local [0.09026009]
Right: 0.0426179604261796
```



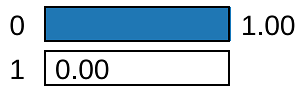
| Feature | Value |
|---|---|
| Origin | 0.44 |
| pressure_msl_dst | 0.65 |

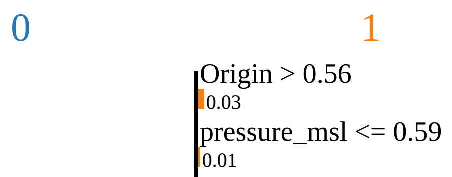
| | |
|---|---|
| Flight | 0.28 |
| relative_humidity_2m | 0.93 |
| Airline | 0.03 |
| wind_speed_10m_dst | 0.15 |
| temperature_2m_dst | 0.43 |
| IATA code | 0.32 |
| precipitation | 0.01 |

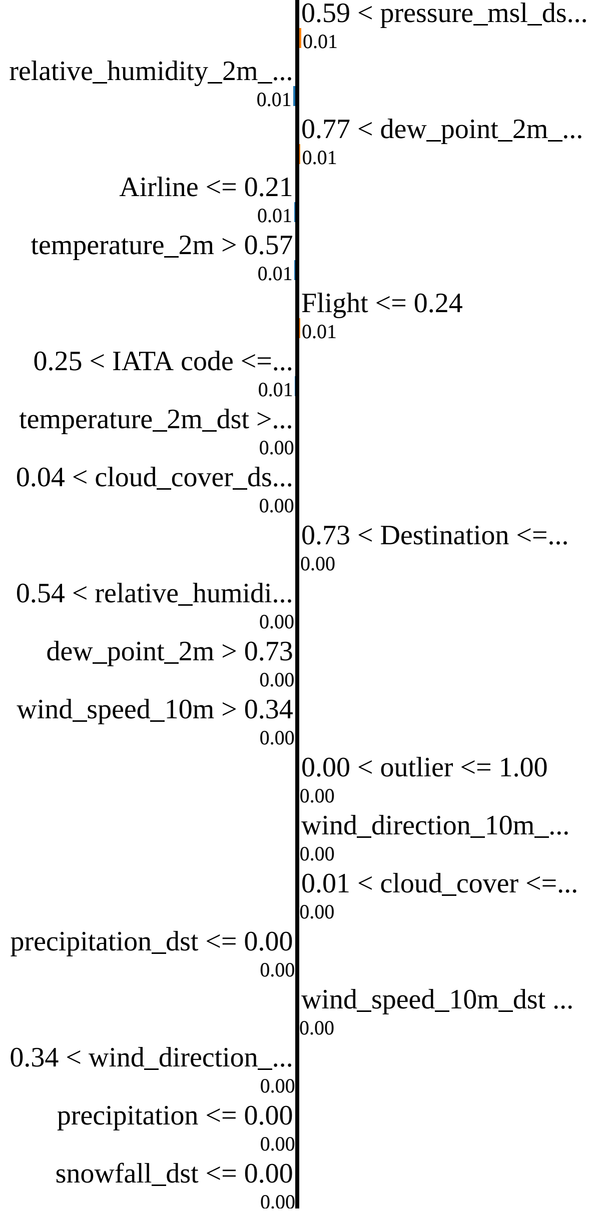
```
Índice: 13051
Intercept 0.09849439014338977
Prediction_local [0.12708042]
Right: 0.0
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

Prediction probabilities



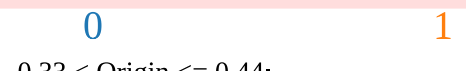


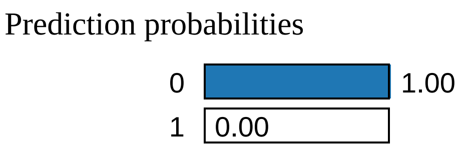


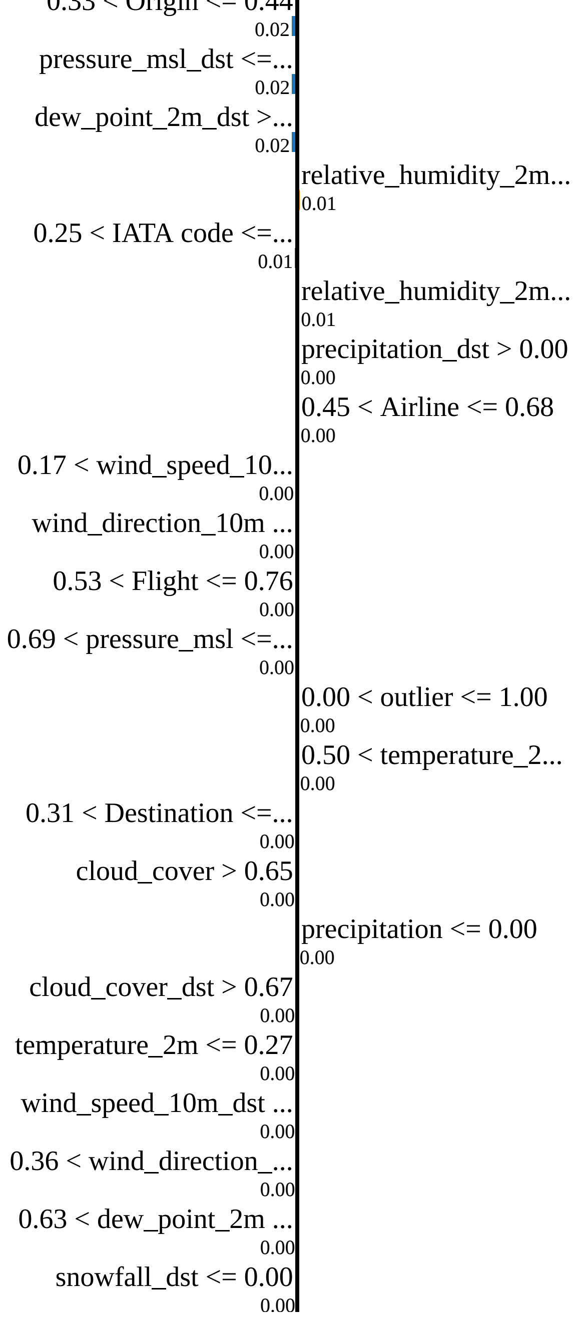
| Feature | Value |
|---|---|
| Origin | 0.67 |
| pressure_msl | 0.43 |
| pressure_msl_dst | 0.61 |
| relative_humidity_2m_dst | 0.49 |
| dew_point_2m_dst | 0.82 |
| Airline | 0.12 |
| temperature_2m | 0.71 |
| Flight | 0.24 |
| IATA code | 0.35 |
| temperature_2m_dst | 0.79 |

```
Índice: 13527
Intercept 0.11466569481229405
Prediction_local [0.06273512]
Right: 0.0
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
e valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```







| Feature | Value |
|---|---|
| Origin | 0.44 |
| pressure_msl_dst | 0.51 |
| dew_point_2m_dst | 0.90 |
| relative_humidity_2m_dst | 0.99 |
| IATA code | 0.33 |
| relative_humidity_2m | 0.99 |
| precipitation_dst | 0.19 |
| Airline | 0.60 |
| wind_speed_10m | 0.18 |
| wind_direction_10m | 0.11 |

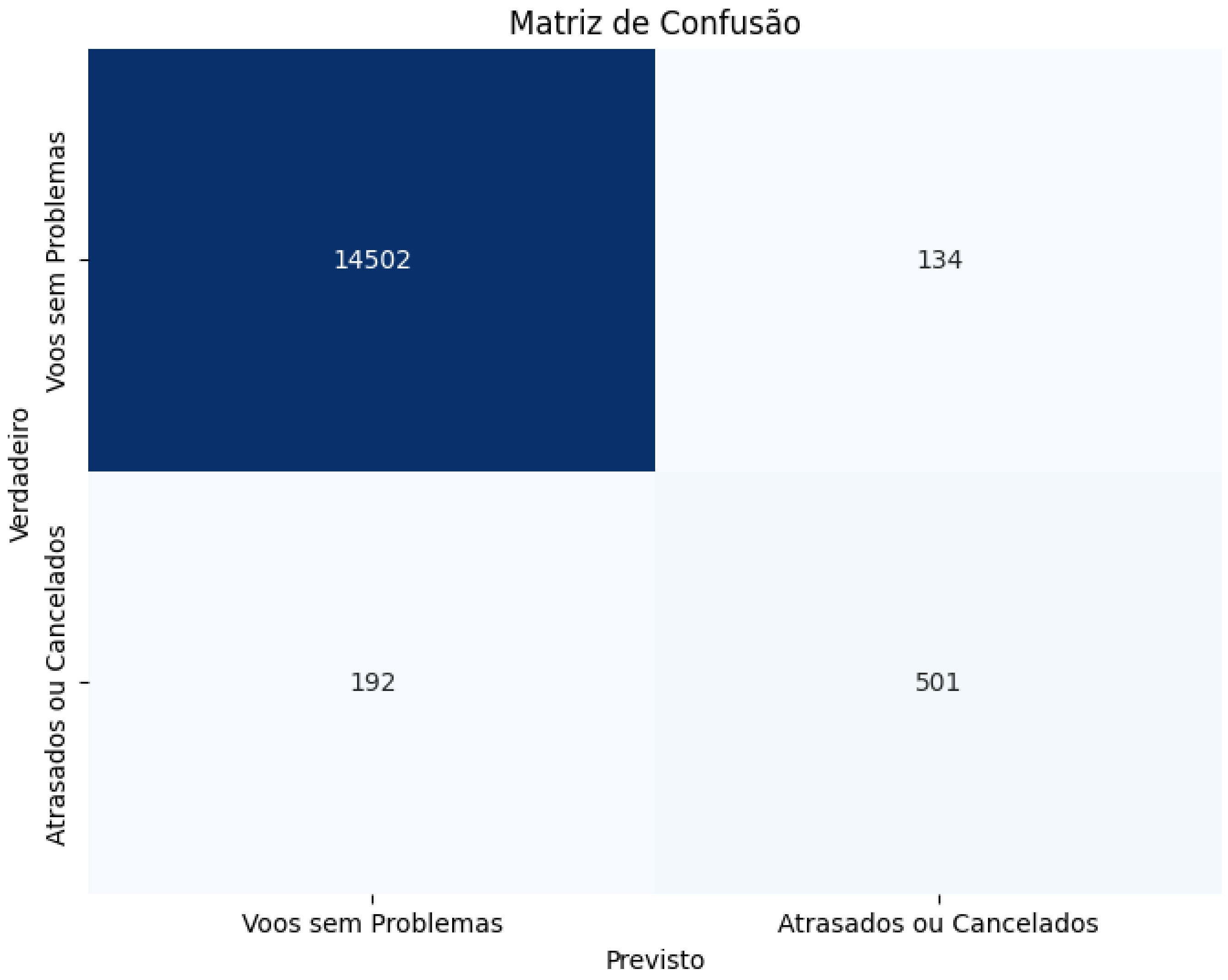
## Experimentos

```
In [28]: def experiment(new_X):
  X_train, X_temp, y_train, y_temp = train_test_split(new_X, y, test_size=0.3, random_st
  X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_

  X_train, y_train = oversampler.fit_resample(X_train, y_train)
  X_val, y_val = oversampler.fit_resample(X_val, y_val)

  model = RandomForestClassifier(n_estimators=18, max_depth=20, random_state=42)
  model.fit(X_train, y_train)

  y_pred = model.predict(X_test)
  test_accuracy = accuracy_score(y_test, y_pred)

  print(f"Acurácia no conjunto de teste: {test_accuracy}")
  print(classification_report(y_test, y_pred))

  cm = confusion_matrix(y_test, y_pred)
  plt.figure(figsize=(8, 6))
  sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
              xticklabels=['Voos sem Problemas', 'Atrasados ou Cancelados'], yticklabels
  plt.xlabel('Previsto')
  plt.ylabel('Verdadeiro')
  plt.title(f'Matriz de Confusão')
  plt.show()
```

- **Tirar colunas de menor relevância**

In [37]:
```
X_test1 = X.drop(columns=['outlier', 'precipitation_dst', 'precipitation', 'snowfall_dst
print(X_test1.columns)
experiment(X_test1)
```

```
Index(['IATA code', 'Destination', 'Flight', 'Origin', 'temperature_2m',
       'relative_humidity_2m', 'dew_point_2m', 'pressure_msl', 'cloud_cover',
       'wind_speed_10m', 'wind_direction_10m', 'temperature_2m_dst',
       'relative_humidity_2m_dst', 'dew_point_2m_dst', 'pressure_msl_dst',
       'wind_speed_10m_dst', 'wind_direction_10m_dst'],
      dtype='object')
Acurácia no conjunto de teste: 0.9787331202296301
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99     14636
         1.0       0.79      0.72      0.75       693

    accuracy                           0.98     15329
   macro avg       0.89      0.86      0.87     15329
weighted avg       0.98      0.98      0.98     15329
```

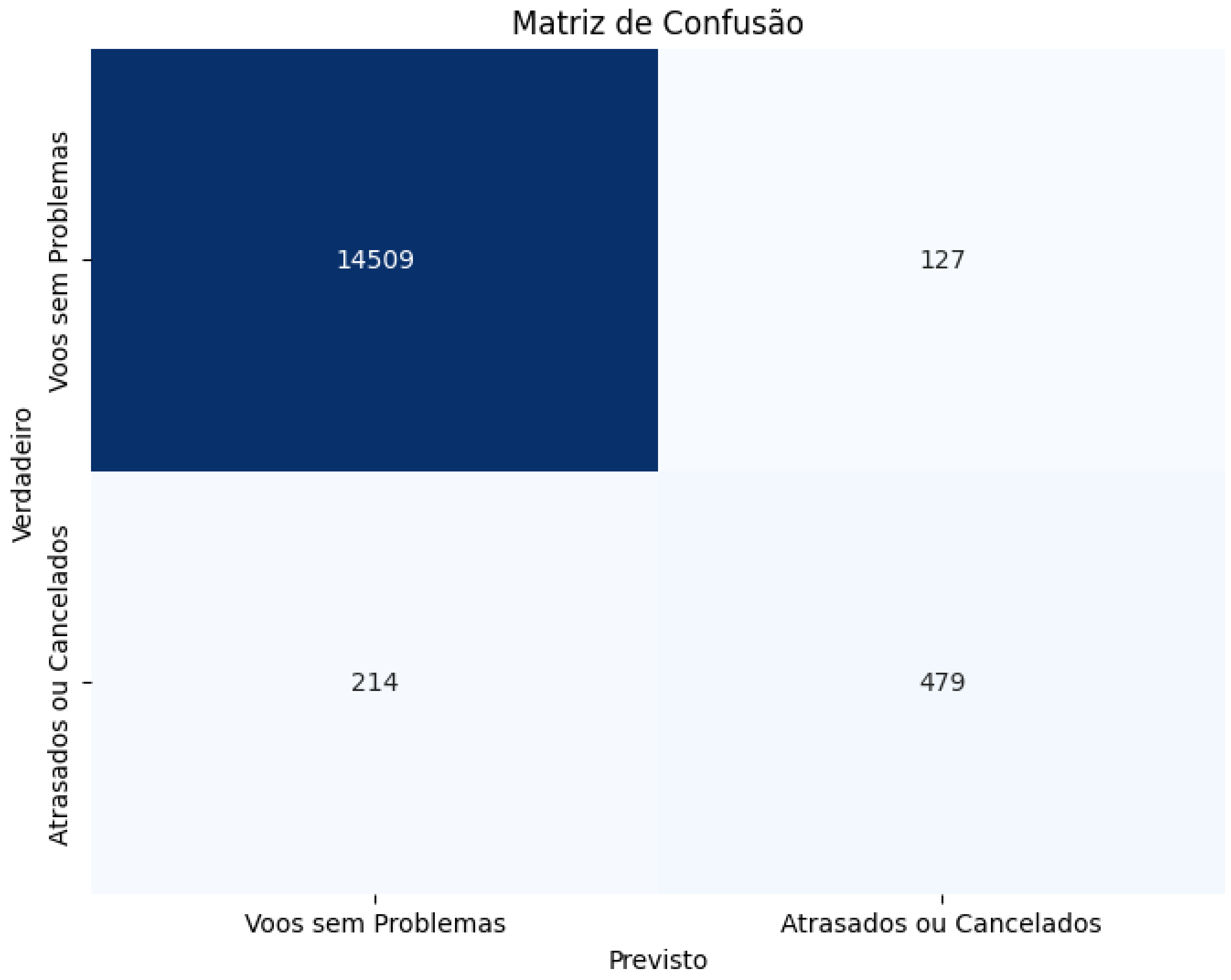


134+192=326.Isso representa uma melhora em comparação ao modelo baseline.

- **Manter apenas as doze melhores colunas**

In [36]:
```python
X_test2 = X[[feature for feature, importance in feature_importance_list[:12]]]
print(X_test2.columns)
experiment(X_test2)
```

```
Index(['dew_point_2m_dst', 'IATA code', 'relative_humidity_2m',
       'wind_direction_10m', 'pressure_msl', 'dew_point_2m', 'Origin',
       'temperature_2m', 'wind_speed_10m', 'pressure_msl_dst',
       'temperature_2m_dst', 'Flight'],
      dtype='object')
Acurácia no conjunto de teste: 0.977754582816883
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99     14636
         1.0       0.79      0.69      0.74       693

    accuracy                           0.98     15329
   macro avg       0.89      0.84      0.86     15329
weighted avg       0.98      0.98      0.98     15329
```

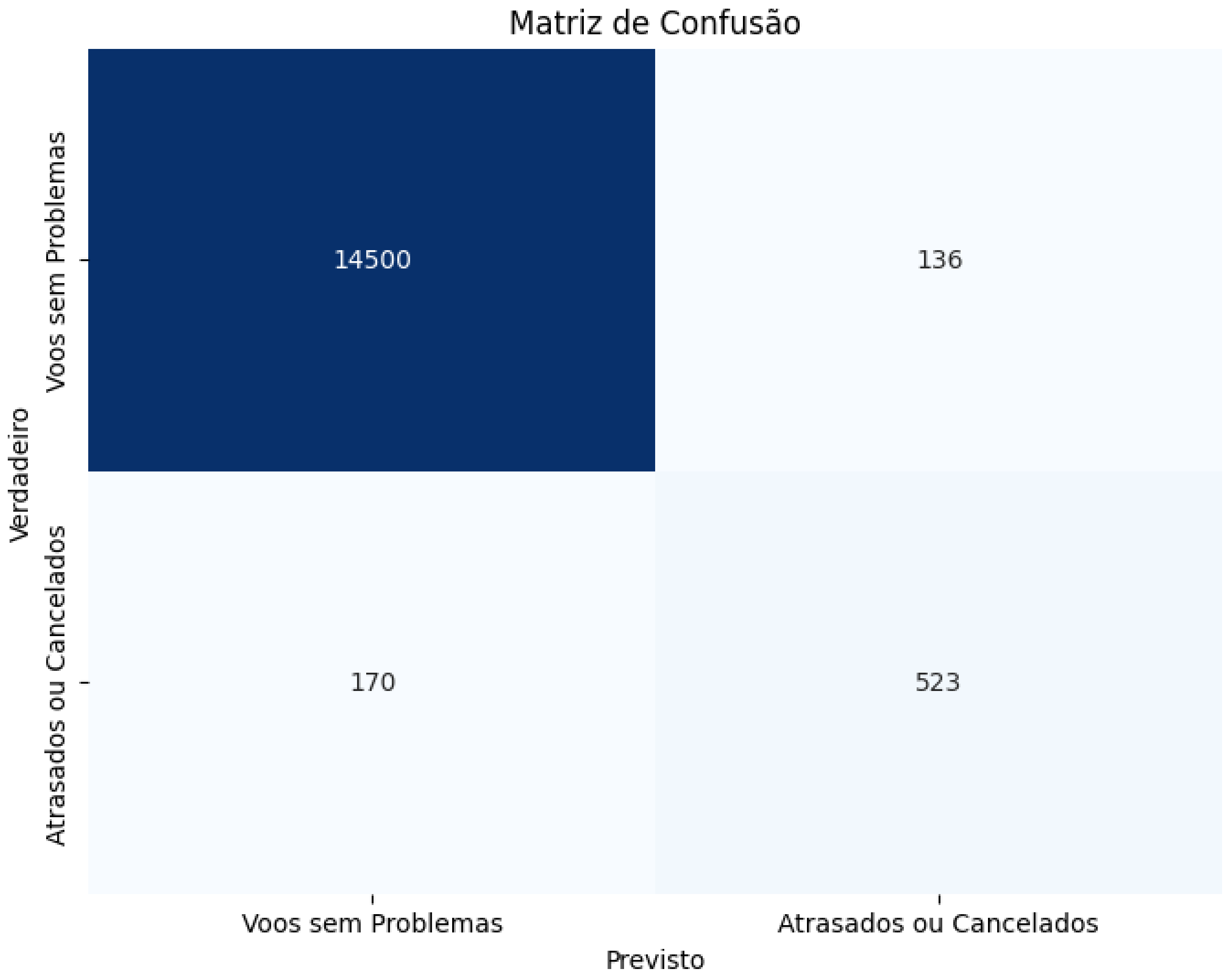


127+214=341. O que significa que houve uma pequena melhora em relação ao baseline, mas um piora em comparação ao experimento 1.

- **Remover todas as colunas com informações sobre o voo**

```
In [38]: X_test3 = X.drop(columns=['IATA code','Destination', 'Flight', 'Airline',  'Origin'])
         print(X_test3.columns)
         experiment(X_test3)
```

```
Index(['temperature_2m', 'relative_humidity_2m', 'dew_point_2m',
       'precipitation', 'pressure_msl', 'cloud_cover', 'wind_speed_10m',
       'wind_direction_10m', 'temperature_2m_dst', 'relative_humidity_2m_dst',
       'dew_point_2m_dst', 'precipitation_dst', 'snowfall_dst',
       'pressure_msl_dst', 'cloud_cover_dst', 'wind_speed_10m_dst',
       'wind_direction_10m_dst', 'outlier'],
      dtype='object')
Acurácia no conjunto de teste: 0.9800378367799596
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99     14636
         1.0       0.79      0.75      0.77       693

    accuracy                           0.98     15329
   macro avg       0.89      0.87      0.88     15329
weighted avg       0.98      0.98      0.98     15329
```

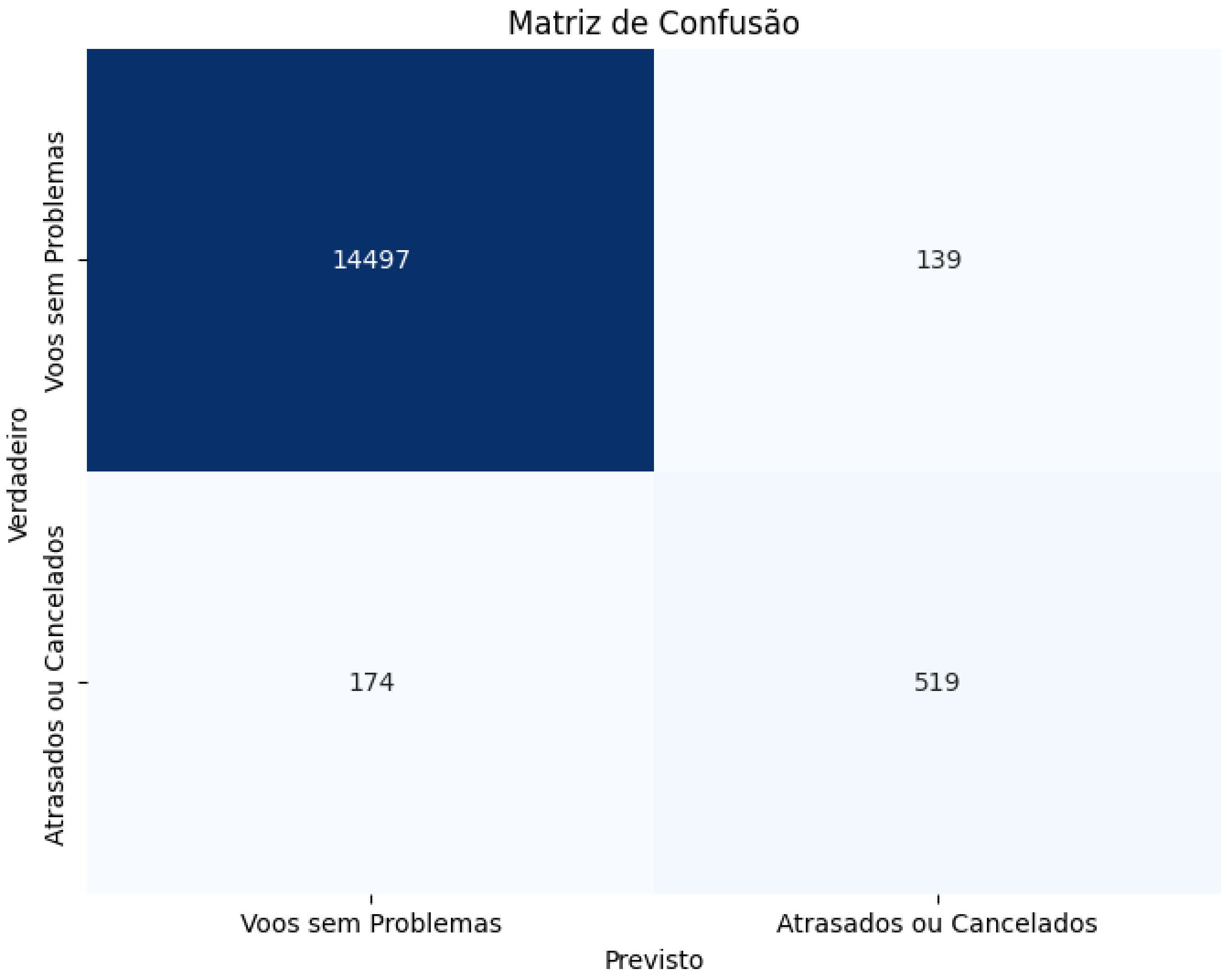


136+170=306. Representa uma melhora significativa, sendo o melhor experimento até o momento.

- **Remover as colunas relacionadas ao voo e as menos importantes**

```
In [42]: X_test4 = X.drop(columns=['IATA code','Destination', 'Flight', 'Airline',  'Origin', 'ou
         print(X_test4.columns)
         experiment(X_test4)
```

```
Index(['temperature_2m', 'relative_humidity_2m', 'dew_point_2m',
       'pressure_msl', 'cloud_cover', 'wind_speed_10m', 'wind_direction_10m',
       'temperature_2m_dst', 'relative_humidity_2m_dst', 'dew_point_2m_dst',
       'pressure_msl_dst', 'wind_speed_10m_dst', 'wind_direction_10m_dst'],
      dtype='object')
Acurácia no conjunto de teste: 0.9795811859873442
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99     14636
         1.0       0.79      0.75      0.77       693

    accuracy                           0.98     15329
   macro avg       0.89      0.87      0.88     15329
weighted avg       0.98      0.98      0.98     15329
```

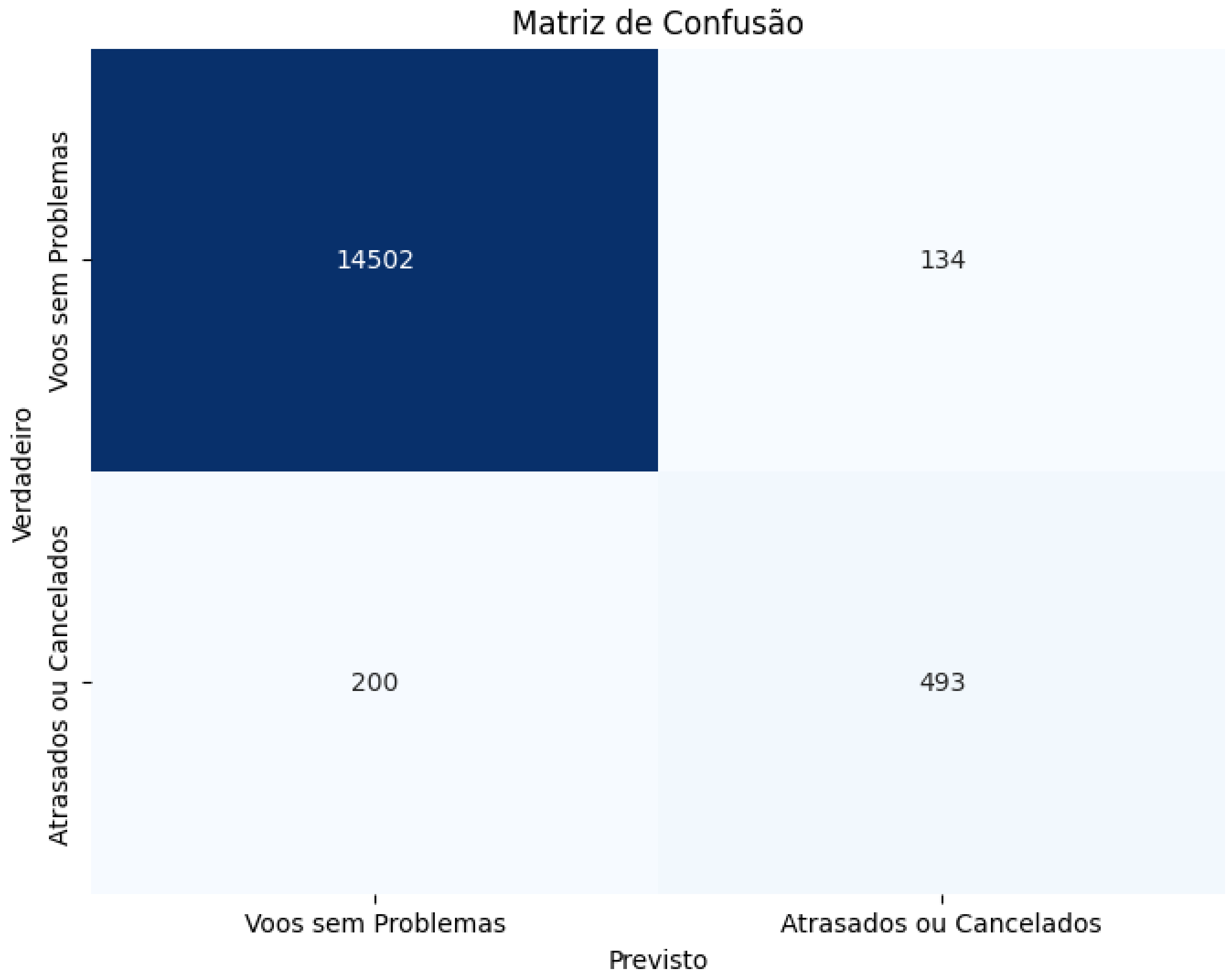


174+139=313. Apesar de ser uma melhora em comparação ao modelo baseline, o experimento 3 apresentou resultados superiores.

- **Remover as colunas relacionadas ao voo, com exceção das que estão entre as 12 melhores, e as menos importantes**

In [43]:
```
X_test5 = X.drop(columns=['Destination', 'Airline', 'outlier', 'precipitation_dst', 'pre
print(X_test5.columns)
experiment(X_test5)
```

```
Index(['IATA code', 'Flight', 'Origin', 'temperature_2m',
       'relative_humidity_2m', 'dew_point_2m', 'pressure_msl', 'cloud_cover',
       'wind_speed_10m', 'wind_direction_10m', 'temperature_2m_dst',
       'relative_humidity_2m_dst', 'dew_point_2m_dst', 'pressure_msl_dst',
       'wind_speed_10m_dst', 'wind_direction_10m_dst'],
      dtype='object')
Acurácia no conjunto de teste: 0.9782112336094984
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99     14636
         1.0       0.79      0.71      0.75       693

    accuracy                           0.98     15329
   macro avg       0.89      0.85      0.87     15329
weighted avg       0.98      0.98      0.98     15329
```

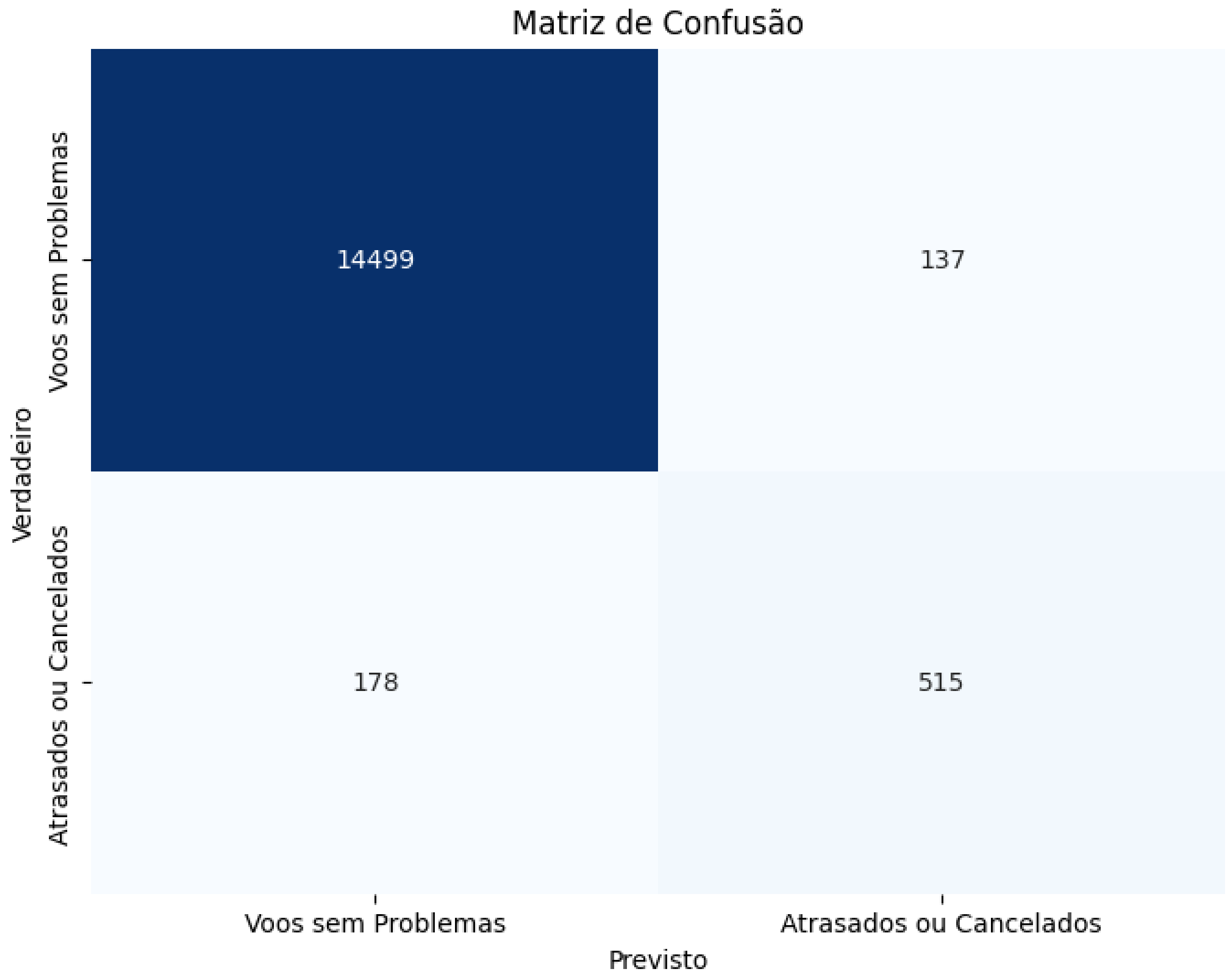


200+134=334. Não houve melhora em relação ao experimento 3.

- **Remover as colunas relacionadas ao voo e as duas menos importantes**

```
In [44]: X_test6 = X.drop(columns=['IATA code','Destination', 'Flight', 'Airline',  'Origin', 'ou
         print(X_test6.columns)
         experiment(X_test6)
```

```
Index(['temperature_2m', 'relative_humidity_2m', 'dew_point_2m',
       'precipitation', 'pressure_msl', 'cloud_cover', 'wind_speed_10m',
       'wind_direction_10m', 'temperature_2m_dst', 'relative_humidity_2m_dst',
       'dew_point_2m_dst', 'precipitation_dst', 'pressure_msl_dst',
       'cloud_cover_dst', 'wind_speed_10m_dst', 'wind_direction_10m_dst'],
      dtype='object')
Acurácia no conjunto de teste: 0.9794507143323113
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99     14636
         1.0       0.79      0.74      0.77       693

    accuracy                           0.98     15329
   macro avg       0.89      0.87      0.88     15329
weighted avg       0.98      0.98      0.98     15329
```

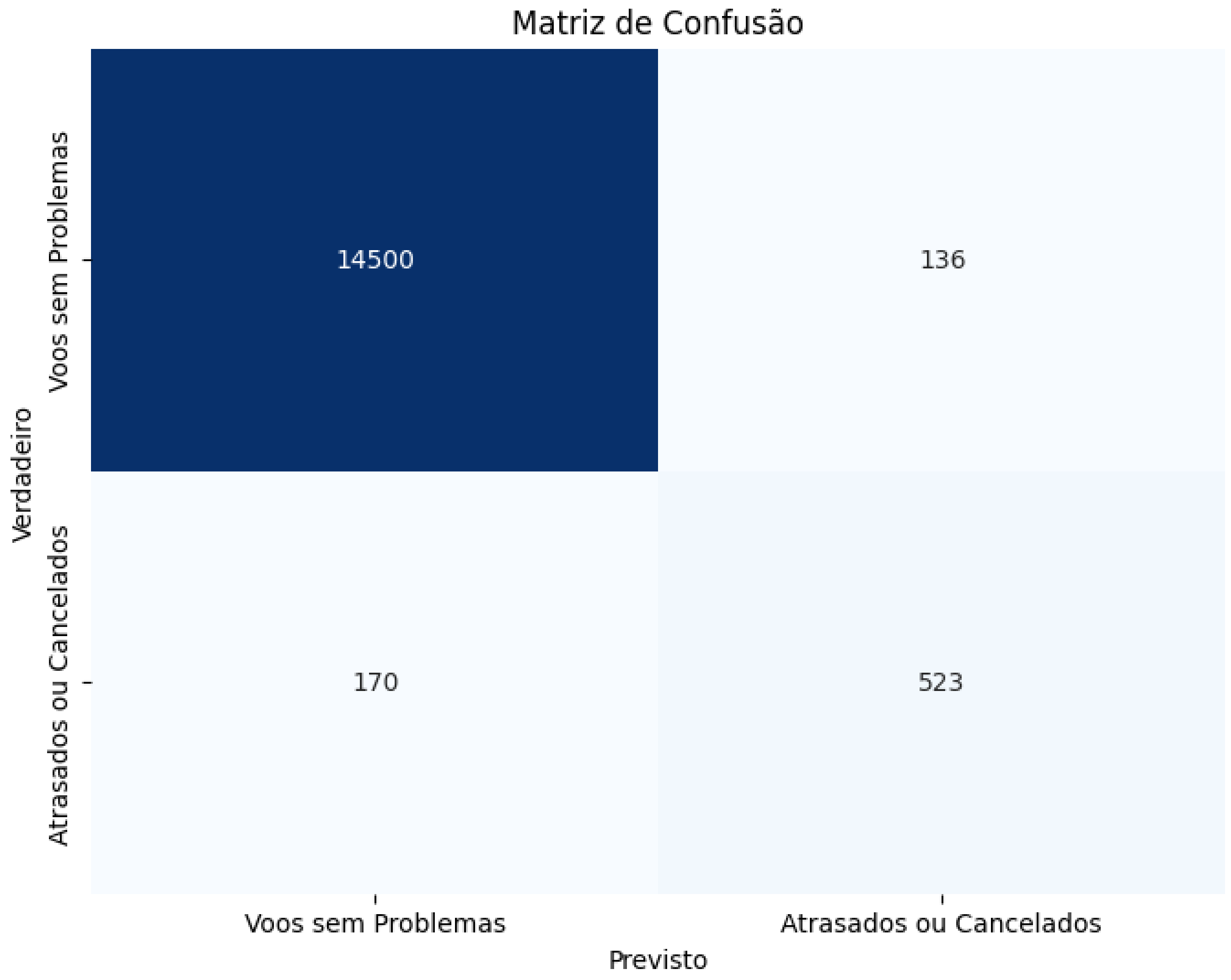


178+137=315. Não houve melhora em relação ao experimento 3.

## Melhor cenário é remover as colunas relacionadas ao voo

In [45]:
```
X_test3 = X.drop(columns=['IATA code','Destination', 'Flight', 'Airline',  'Origin'])
print(X_test3.columns)
experiment(X_test3)
```

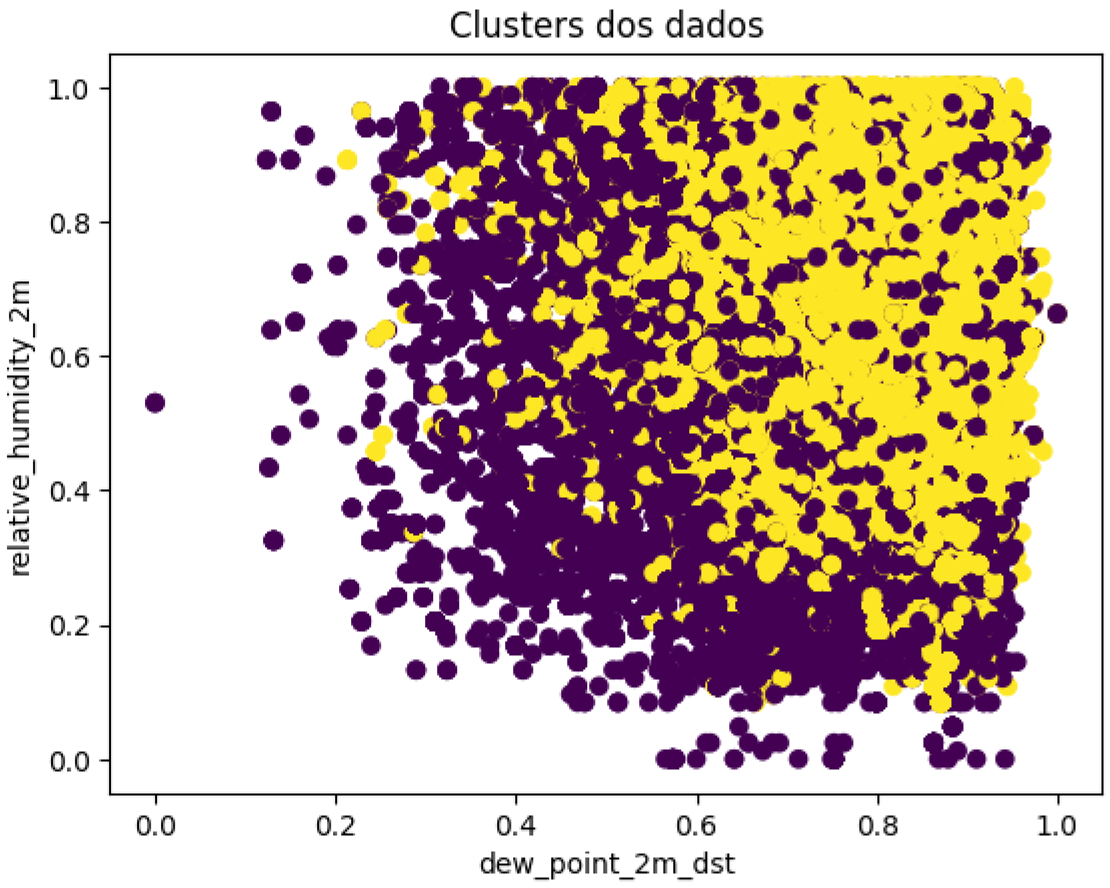
```
Index(['temperature_2m', 'relative_humidity_2m', 'dew_point_2m',
       'precipitation', 'pressure_msl', 'cloud_cover', 'wind_speed_10m',
       'wind_direction_10m', 'temperature_2m_dst', 'relative_humidity_2m_dst',
       'dew_point_2m_dst', 'precipitation_dst', 'snowfall_dst',
       'pressure_msl_dst', 'cloud_cover_dst', 'wind_speed_10m_dst',
       'wind_direction_10m_dst', 'outlier'],
      dtype='object')
Acurácia no conjunto de teste: 0.9800378367799596
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99     14636
         1.0       0.79      0.75      0.77       693

    accuracy                           0.98     15329
   macro avg       0.89      0.87      0.88     15329
weighted avg       0.98      0.98      0.98     15329
```



## Bônus: Clustering para entendimento dos dados

In [59]:
```
# Clusterização com KMeans
kmeans = KMeans(n_clusters=2, random_state=42)
clusters_kmeans = kmeans.fit_predict(X_train)

plt.scatter(X_train['dew_point_2m_dst'], X_train['relative_humidity_2m'], c=clusters_kme
plt.title('Clusters dos dados')
plt.xlabel('dew_point_2m_dst')
plt.ylabel('relative_humidity_2m')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: T
he default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_i
nit` explicitly to suppress the warning
  warnings.warn(
```


Clusters dos dados
relative_humidity_2m
dew_point_2m_dst
0.0
0.2
0.4
0.6
0.8
1.0