##AIDI 1002 Final Term Project Report

Student name: Krunalkumar Solanki(200583826)

Email:200583826@student.georgianc.on.ca (mailto:200583826@student.georgianc.on.ca)

Student name: Rohan Savaliya(200544272)

Email: 200544272@student.georgianc.on.ca (mailto:200544272@student.georgianc.on.ca)

Research Paper Link: https://paperswithcode.com/task/time-series-forecasting (https://paperswithcode.com/task/time-series-forecasting)

Introduction:

Time series forecasting is a predictive modeling technique essential for understanding and predicting future trends based on past observations collected over time. It involves analyzing sequential data points to uncover underlying patterns, such as trends, seasonality, and cyclical behavior, which influence the target variable's evolution. Various forecasting models, ranging from traditional statistical methods like ARIMA and exponential smoothing to advanced machine learning algorithms such as neural networks and ensemble techniques, are employed depending on the data characteristics and forecasting requirements. The forecasting process typically involves data preprocessing, model selection, training, evaluation, and generating forecasts for future time points. Challenges such as dealing with noisy data, ensuring stationarity, selecting appropriate model parameters, and incorporating external factors are addressed during the forecasting process. Ultimately, time series forecasting empowers businesses and researchers to make informed decisions, anticipate market trends, and optimize resource allocation strategies based on historical data patterns and future predictions.

Time series forecasting plays a crucial role in guiding immediate decision-making processes. It provides valuable insights into near-future trends, allowing businesses to adjust their strategies promptly. By accurately predicting short-term fluctuations in demand, sales, or resource utilization, organizations can optimize inventory levels, allocate resources efficiently, manage cash flow effectively, and enhance operational performance. Time series forecasting also aids in risk mitigation by identifying potential anomalies or deviations from expected patterns, enabling proactive interventions to minimize disruptions and capitalize on emerging opportunities. In essence, in the short term, time series forecasting serves as a critical tool for agility, responsiveness, and proactive decision-making in dynamic and competitive environments.

Aim:

Our primary objective is to compare these models comprehensively. We aim to identify the most effective approach for accurate and efficient time series predictions. By evaluating their performance, we can guide practitioners toward the best choice for their specific forecasting tasks. Time series forecasting serves as a pivotal tool for businesses aiming to navigate the complexities of dynamic markets and uncertain future scenarios. Its primary aim lies in accurately predicting future trends and patterns based on historical data, enabling

organizations to anticipate changes and make well-informed decisions. By leveraging sophisticated modeling techniques, businesses can forecast short-term and long-term outcomes, facilitating effective planning, resource allocation, and risk management strategies. Additionally, time series forecasting aids in optimizing operational efficiency and enhancing overall performance by providing valuable insights into demand fluctuations, market dynamics, and emerging opportunities.

Furthermore, the aim extends beyond mere prediction, encompassing decision support and continuous improvement initiatives. Time series forecasting equips decision-makers with actionable insights to respond proactively to evolving market conditions, mitigate risks, and capitalize on emerging trends. Through rigorous evaluation and refinement of forecasting models, organizations can enhance the accuracy and reliability of their predictions, driving better decision-making and sustained competitive advantage in today's dynamic business landscape.

Github Repo:https://github.com/unit8co/darts/tree/master (https://github.com/unit8co/darts/tree/master)

Discription of the paper: Our paper delves into the following aspects:

Model Overview: ARIMA: We discuss the classical ARIMA model, which combines autoregressive, differencing, and moving average components. ARIMA serves as a baseline for comparison.

LSTM: Long Short-Term Memory networks, a type of recurrent neural network (RNN), excel at capturing complex temporal dependencies. We explore their architecture and strengths. Prophet: Developed by Facebook, Prophet is designed for time series forecasting. It incorporates seasonality, holidays, and trend components.

Motivation: We chose ARIMA as a traditional statistical method due to its historical significance and simplicity. LSTM represents the deep learning paradigm, capable of handling intricate patterns. Prophet bridges the gap between statistical models and deep learning, offering robustness and ease of use.

Dataset: For this we use two datasets which is air passanger dataset and monthlymilk dataset.

PROBLEM STATEMENT :

Time series forecasting stands as a linchpin in the decision-making process across various real-world scenarios, offering invaluable insights into future events. Its significance unfolds in multiple domains, starting with resource allocation optimization, where precise forecasts guide efficient allocation of budgets, personnel, and equipment, averting overinvestment or shortages. Inventory planning hinges on accurate predictions to strike a balance between supply and demand, ensuring optimal stock levels without tying up excess capital. Moreover, time series forecasting underpins critical business decisions, from pricing strategies to marketing campaigns, enabling organizations to adapt and thrive in dynamic markets. In the realm of risk management, forecasting helps mitigate uncertainties by anticipating market fluctuations, demand shifts, and supply chain disruptions, allowing proactive responses to mitigate potential risks. Operational efficiency, financial planning, and budgeting also benefit from reliable forecasts, driving better resource utilization, strategic planning, and fiscal

management. In essence, the precision of time series forecasting empowers decision-makers to optimize resources, plan effectively, and navigate uncertainties across diverse industries, ultimately leading to better outcomes and informed choices.

CONTEXT OF THE PROBLEM:

The need for accurate time series forecasting arises in scenarios where decision-makers rely on predictions to optimize resource allocation, plan inventory, or make informed business decisions.

SOLUTION:

we use N-Beats model. The N-BEATS (Neural Basis Expansion Analysis for Time Series Forecasting) model is a powerful tool for predicting future trends in time series data. It utilizes deep learning architecture to capture complex temporal patterns without relying on explicit data assumptions. By decomposing time series into interpretable components like trends and seasonality, N-BEATS offers insights while scaling well to different types of data and forecasting horizons. When training, practitioners commonly use loss functions like MAE or MSE and optimization algorithms such as Adam or RMSProp. Despite its strengths, challenges include managing data preprocessing and model complexity. Overall, N-BEATS presents a versatile and interpretable solution for accurate time series forecasting.

Background

Reference	Explanation	Dataset/Input	Weakness
ARIMA	ARIMA assumes stationarity, which may not hold for all time series data.		
LSTM	Requires large amounts of data and longer training times.		
Prophet	May not handle irregularly spaced data well.		
N-BEATS	The dataset includes both univariate (individual stock prices) and multivariate (multiple stock prices) time series.		N-BEATS is computationally expensive due to its deep architecture with multiple stacks and blocks.

Implement paper code :

For the AirPassenger dataset, logistic regression was employed, a method commonly used for binary classification tasks. Logistic regression models the relationship between features in the dataset and predicts a binary outcome, making it suitable for scenarios where the dependent variable is categorical. On the other hand, for the MonthlyMilk dataset, you applied a random forest model. Random forest is an ensemble learning method known for its robustness and ability to handle complex datasets. By building multiple decision trees during training and aggregating their predictions, random forest is adept at capturing intricate patterns in the data. Both approaches have been tailored to the specific characteristics of their respective datasets,

In [ ]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances_argmin_min
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import warnings

warnings.filterwarnings("ignore")
AP = pd.read_csv('air_passengers.csv')
AP.head()
```

Out[1]:

	Month	#Passengers
0	1949-01	112
1	1949-02	118
2	1949-03	132
3	1949-04	129
4	1949-05	121

In [ ]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

X = AP[['Month']]
y = AP['#Passengers']

X['Month'] = pd.to_datetime(X['Month'])


X['Year'] = X['Month'].dt.year
X['Month'] = X['Month'].dt.month


AP.dropna(inplace=True)

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc

# Creating linear regression model
model = LinearRegression()

# Training the model
model.fit(X_train, y_train)

# Making predictions on the testing set
predictions = model.predict(X_test)

# Calculating Mean Squared Error (MSE)
mse = mean_squared_error(y_test, predictions)
print("Mean Squared Error:", mse)

# Optionally, you can also print the model coefficients
print("Model Coefficients:", model.coef_)

# Optionally, you can also print the model intercept
print("Model Intercept:", model.intercept_)
```

```
Mean Squared Error: 1564.4397463792552
Model Coefficients: [ 2.3552502  32.80411101]
Model Intercept: -63848.95598591588
```

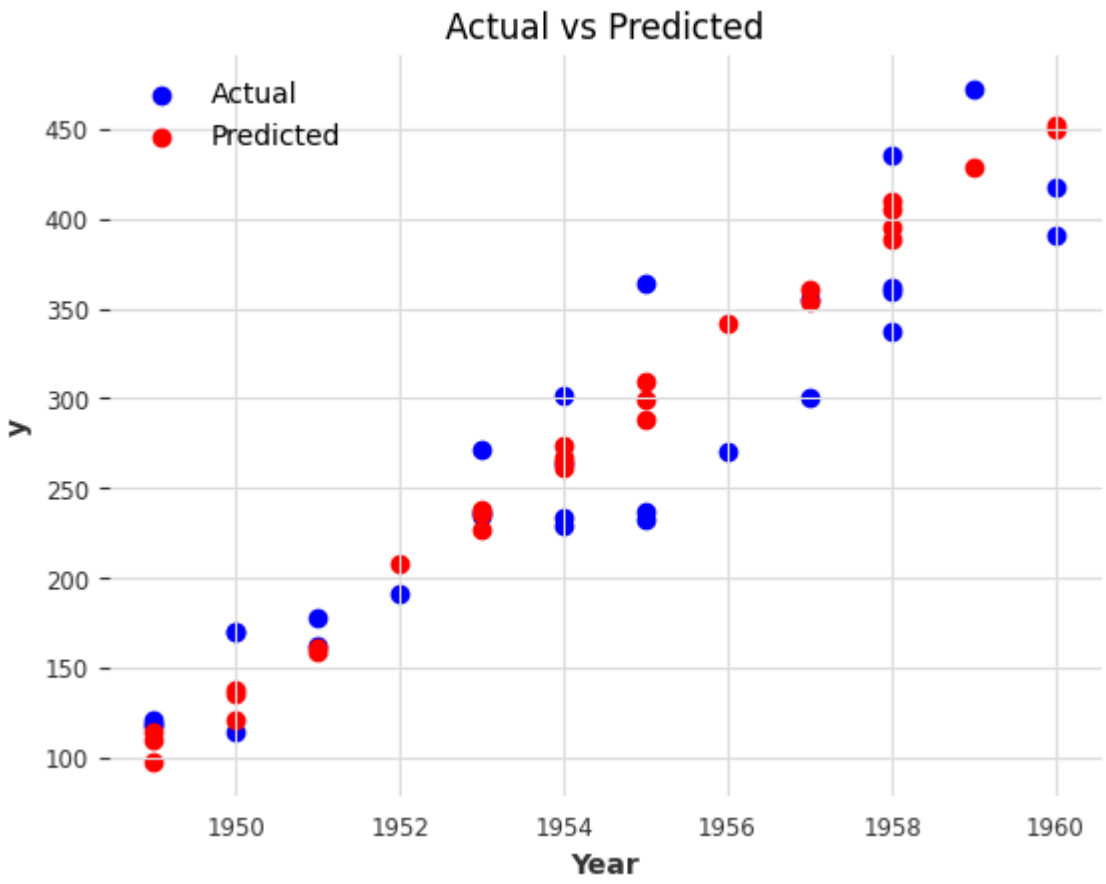
In [ ]:
```
import matplotlib.pyplot as plt

# Plotting actual values vs. predicted values
plt.scatter(X_test['Year'], y_test, color='blue', label='Actual')
plt.scatter(X_test['Year'], predictions, color='red', label='Predicted')

# Adding labels and title
plt.xlabel('Year')
plt.ylabel('y')
plt.title('Actual vs Predicted')

# Adding legend
plt.legend()

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



In [ ]:

```
# Importing necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

X = AP.drop(columns=['#Passengers'])  # Features
y = AP['#Passengers']                  # Target variable

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc

# Define categorical and numerical columns
categorical_cols = [col for col in X.columns if X[col].dtype == 'object']
numerical_cols = [col for col in X.columns if X[col].dtype in ['int64', 'float

# Preprocess categorical columns: one-hot encode
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols),
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_cols)
    ],
    remainder='passthrough'
)

# Fit preprocessor and transform data
X_train_processed = preprocessor.fit_transform(X_train)
X_test_processed = preprocessor.transform(X_test)

# Creating logistic regression model
model = LogisticRegression()

# Training the model
model.fit(X_train_processed, y_train)

# Making predictions on the testing set
predictions = model.predict(X_test_processed)

# Calculating accuracy
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

# Printing classification report
print("Classification Report:")
print(classification_report(y_test, predictions))
```

Accuracy: 0.0
Classification Report:

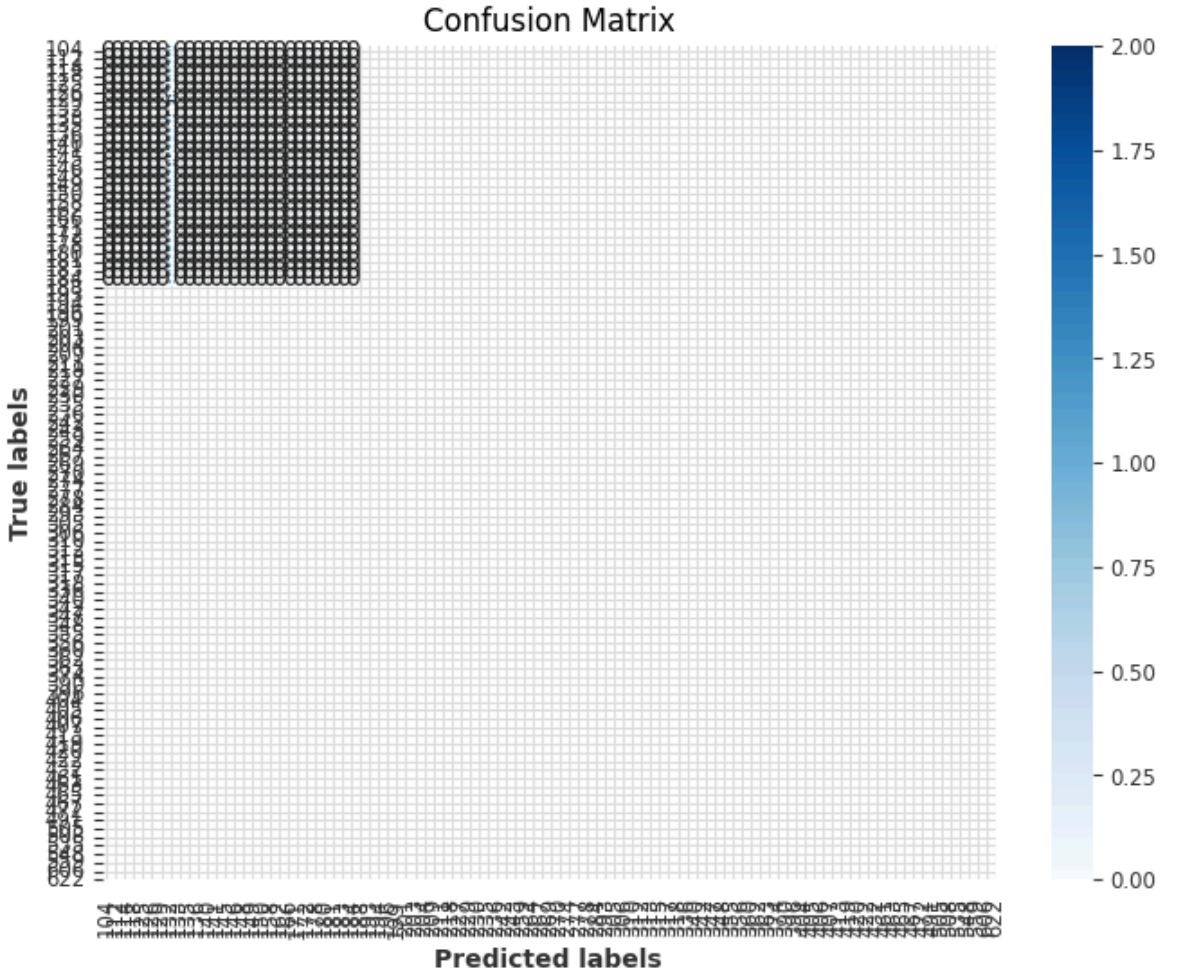
	precision	recall	f1-score	support
115	0.00	0.00	0.00	1.0
118	0.00	0.00	0.00	1.0
119	0.00	0.00	0.00	1.0
121	0.00	0.00	0.00	1.0
163	0.00	0.00	0.00	1.0
170	0.00	0.00	0.00	2.0
178	0.00	0.00	0.00	1.0
180	0.00	0.00	0.00	0.0
191	0.00	0.00	0.00	1.0
229	0.00	0.00	0.00	1.0
233	0.00	0.00	0.00	1.0
234	0.00	0.00	0.00	1.0
235	0.00	0.00	0.00	1.0
237	0.00	0.00	0.00	2.0
264	0.00	0.00	0.00	1.0
271	0.00	0.00	0.00	1.0
272	0.00	0.00	0.00	1.0
301	0.00	0.00	0.00	1.0
302	0.00	0.00	0.00	1.0
337	0.00	0.00	0.00	1.0
355	0.00	0.00	0.00	1.0
359	0.00	0.00	0.00	1.0
362	0.00	0.00	0.00	1.0
364	0.00	0.00	0.00	1.0
391	0.00	0.00	0.00	1.0
417	0.00	0.00	0.00	1.0
435	0.00	0.00	0.00	1.0
472	0.00	0.00	0.00	1.0
accuracy			0.00	29.0
macro avg	0.00	0.00	0.00	29.0
weighted avg	0.00	0.00	0.00	29.0

In [ ]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Calculate confusion matrix
conf_matrix = confusion_matrix(y_test, predictions)

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=model.classes_, yticklabels=model.classes_)
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
```



In [ ]:
```
MD = pd.read_csv('MonthlyMilkDataset.csv')
MD.head()
```

Out[19]:

	Month	Pounds per cow
0	1962-01	589
1	1962-02	561
2	1962-03	640
3	1962-04	656
4	1962-05	727

In [ ]:

```
# Importing necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer



# Assuming your CSV file has features and a target variable
X = MD.drop(columns=['Pounds per cow'])  # Features
y = MD['Pounds per cow']                  # Target variable

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, randc

# Define categorical and numerical columns
categorical_cols = [col for col in X.columns if X[col].dtype == 'object']
numerical_cols = [col for col in X.columns if X[col].dtype in ['int64', 'float

# Preprocess numerical columns: impute missing values and scale
numerical_transformer = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_cols)
    ],
    remainder='passthrough'
)

# Preprocess categorical columns: impute missing values and one-hot encode
categorical_transformer = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_cols)
    ],
    remainder='passthrough'
)

# Combine preprocessing steps
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)
    ]
)

# Fit preprocessor and transform data
X_train_processed = preprocessor.fit_transform(X_train)
X_test_processed = preprocessor.transform(X_test)

# Creating Random Forest classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Training the model
model.fit(X_train_processed, y_train)

# Making predictions on the testing set
```

```python
predictions = model.predict(X_test_processed)

# Calculating accuracy
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

# Printing classification report
print("Classification Report:")
print(classification_report(y_test, predictions))
```

```
Accuracy: 0.0
Classification Report:
              precision    recall  f1-score   support

         577       0.00      0.00      0.00       1.0
         600       0.00      0.00      0.00       1.0
         617       0.00      0.00      0.00       1.0
         621       0.00      0.00      0.00       1.0
         628       0.00      0.00      0.00       1.0
         639       0.00      0.00      0.00       1.0
         653       0.00      0.00      0.00       0.0
         660       0.00      0.00      0.00       1.0
         661       0.00      0.00      0.00       1.0
         673       0.00      0.00      0.00       1.0
         677       0.00      0.00      0.00       1.0
         678       0.00      0.00      0.00       1.0
         681       0.00      0.00      0.00       1.0
         690       0.00      0.00      0.00       1.0
         697       0.00      0.00      0.00       1.0
         702       0.00      0.00      0.00       1.0
         713       0.00      0.00      0.00       1.0
         736       0.00      0.00      0.00       1.0
         742       0.00      0.00      0.00       1.0
         747       0.00      0.00      0.00       1.0
         755       0.00      0.00      0.00       1.0
         756       0.00      0.00      0.00       1.0
         760       0.00      0.00      0.00       1.0
         773       0.00      0.00      0.00       1.0
         796       0.00      0.00      0.00       1.0
         800       0.00      0.00      0.00       1.0
         804       0.00      0.00      0.00       1.0
         807       0.00      0.00      0.00       1.0
         817       0.00      0.00      0.00       1.0
         834       0.00      0.00      0.00       1.0
         858       0.00      0.00      0.00       1.0
         878       0.00      0.00      0.00       1.0
         924       0.00      0.00      0.00       1.0
         937       0.00      0.00      0.00       1.0
         969       0.00      0.00      0.00       1.0

    accuracy                           0.00      34.0
   macro avg       0.00      0.00      0.00      34.0
weighted avg       0.00      0.00      0.00      34.0
```

In [ ]:
```
! pip install pdpbox
```

```
Collecting pdpbox
  Downloading PDPbox-0.3.0-py3-none-any.whl (35.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 35.8/35.8 MB 25.0 MB/s eta 0:0
0:00
Requirement already satisfied: joblib>=1.1.0 in /usr/local/lib/python3.10/di
st-packages (from pdpbox) (1.4.0)
Requirement already satisfied: matplotlib>=3.6.2 in /usr/local/lib/python3.1
0/dist-packages (from pdpbox) (3.7.1)
Requirement already satisfied: numpy>=1.21.5 in /usr/local/lib/python3.10/di
st-packages (from pdpbox) (1.25.2)
Requirement already satisfied: pandas>=1.4.4 in /usr/local/lib/python3.10/di
st-packages (from pdpbox) (2.0.3)
Requirement already satisfied: plotly>=5.9.0 in /usr/local/lib/python3.10/di
st-packages (from pdpbox) (5.15.0)
Collecting pqdm>=0.2.0 (from pdpbox)
  Downloading pqdm-0.2.0-py2.py3-none-any.whl (6.8 kB)
Requirement already satisfied: psutil>=5.9.0 in /usr/local/lib/python3.10/di
st-packages (from pdpbox) (5.9.5)
Requirement already satisfied: pytest in /usr/local/lib/python3.10/dist-pack
ages (from pdpbox) (7.4.4)
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python
3.10/dist-packages (from pdpbox) (1.2.2)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-
packages (from pdpbox) (67.7.2)
Requirement already satisfied: sphinx>=5.0.2 in /usr/local/lib/python3.10/di
st-packages (from pdpbox) (5.0.2)
Collecting sphinx-rtd-theme>=1.1.1 (from pdpbox)
  Downloading sphinx_rtd_theme-2.0.0-py2.py3-none-any.whl (2.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.8/2.8 MB 78.4 MB/s eta 0:00:
00
Requirement already satisfied: tqdm>=4.64.1 in /usr/local/lib/python3.10/dis
t-packages (from pdpbox) (4.66.2)
Collecting numpydoc>=1.4.0 (from pdpbox)
  Downloading numpydoc-1.7.0-py3-none-any.whl (62 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.8/62.8 kB 7.7 MB/s eta 0:0
0:00
Requirement already satisfied: xgboost>=1.7.1 in /usr/local/lib/python3.10/d
ist-packages (from pdpbox) (2.0.3)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib>=3.6.2->pdpbox) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dis
t-packages (from matplotlib>=3.6.2->pdpbox) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib>=3.6.2->pdpbox) (4.51.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib>=3.6.2->pdpbox) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/
dist-packages (from matplotlib>=3.6.2->pdpbox) (24.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/di
st-packages (from matplotlib>=3.6.2->pdpbox) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib>=3.6.2->pdpbox) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python
3.10/dist-packages (from matplotlib>=3.6.2->pdpbox) (2.8.2)
Collecting sphinx>=5.0.2 (from pdpbox)
  Downloading sphinx-7.3.6-py3-none-any.whl (3.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.3/3.3 MB 59.5 MB/s eta 0:00:
```

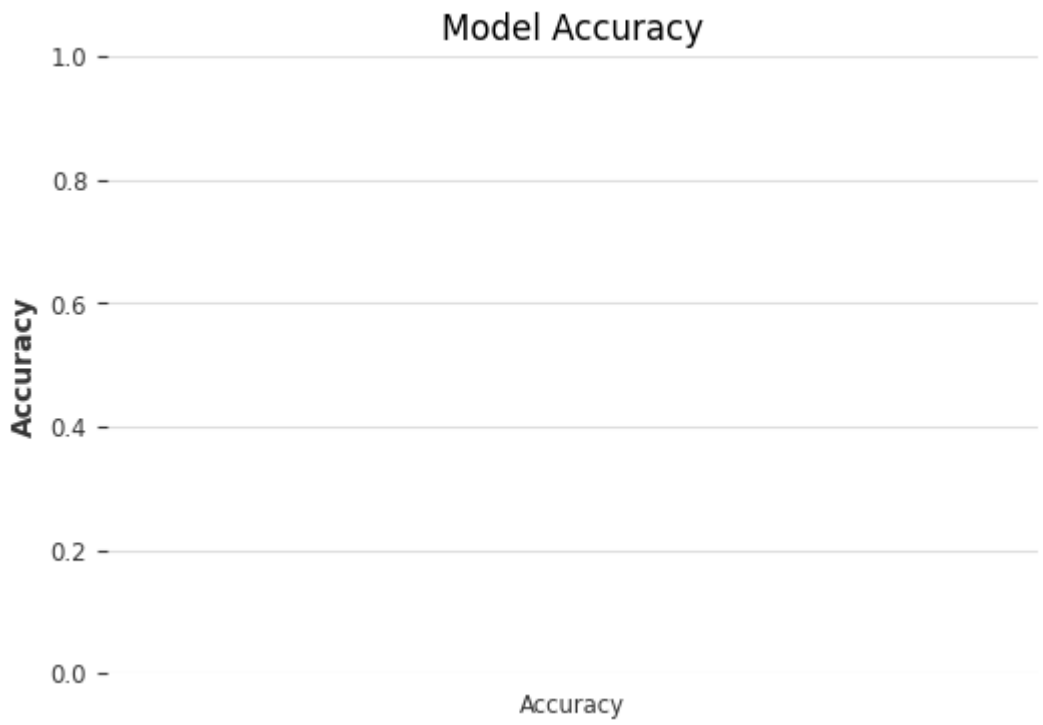
```
00
Requirement already satisfied: tabulate>=0.8.10 in /usr/local/lib/python3.1
0/dist-packages (from numpydoc>=1.4.0->pdpbox) (0.9.0)
Requirement already satisfied: tomli>=1.1.0 in /usr/local/lib/python3.10/dis
t-packages (from numpydoc>=1.4.0->pdpbox) (2.0.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dis
t-packages (from pandas>=1.4.4->pdpbox) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/d
ist-packages (from pandas>=1.4.4->pdpbox) (2024.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/
dist-packages (from plotly>=5.9.0->pdpbox) (8.2.3)
Collecting bounded-pool-executor (from pqdm>=0.2.0->pdpbox)
  Downloading bounded_pool_executor-0.0.3-py3-none-any.whl (3.4 kB)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.1
0/dist-packages (from pqdm>=0.2.0->pdpbox) (4.11.0)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dis
t-packages (from scikit-learn>=1.0.2->pdpbox) (1.11.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python
3.10/dist-packages (from scikit-learn>=1.0.2->pdpbox) (3.4.0)
Requirement already satisfied: sphinxcontrib-applehelp in /usr/local/lib/pyt
hon3.10/dist-packages (from sphinx>=5.0.2->pdpbox) (1.0.8)
Requirement already satisfied: sphinxcontrib-devhelp in /usr/local/lib/pytho
n3.10/dist-packages (from sphinx>=5.0.2->pdpbox) (1.0.6)
Requirement already satisfied: sphinxcontrib-jsmath in /usr/local/lib/python
3.10/dist-packages (from sphinx>=5.0.2->pdpbox) (1.0.1)
Requirement already satisfied: sphinxcontrib-htmlhelp>=2.0.0 in /usr/local/l
ib/python3.10/dist-packages (from sphinx>=5.0.2->pdpbox) (2.0.5)
Requirement already satisfied: sphinxcontrib-serializinghtml>=1.1.9 in /usr/
local/lib/python3.10/dist-packages (from sphinx>=5.0.2->pdpbox) (1.1.10)
Requirement already satisfied: sphinxcontrib-qthelp in /usr/local/lib/python
3.10/dist-packages (from sphinx>=5.0.2->pdpbox) (1.0.7)
Requirement already satisfied: Jinja2>=3.0 in /usr/local/lib/python3.10/dist
-packages (from sphinx>=5.0.2->pdpbox) (3.1.3)
Requirement already satisfied: Pygments>=2.14 in /usr/local/lib/python3.10/d
ist-packages (from sphinx>=5.0.2->pdpbox) (2.16.1)
Requirement already satisfied: docutils<0.22,>=0.18.1 in /usr/local/lib/pyth
on3.10/dist-packages (from sphinx>=5.0.2->pdpbox) (0.18.1)
Requirement already satisfied: snowballstemmer>=2.0 in /usr/local/lib/python
3.10/dist-packages (from sphinx>=5.0.2->pdpbox) (2.2.0)
Requirement already satisfied: babel>=2.9 in /usr/local/lib/python3.10/dist-
packages (from sphinx>=5.0.2->pdpbox) (2.14.0)
Requirement already satisfied: alabaster~=0.7.14 in /usr/local/lib/python3.1
0/dist-packages (from sphinx>=5.0.2->pdpbox) (0.7.16)
Requirement already satisfied: imagesize>=1.3 in /usr/local/lib/python3.10/d
ist-packages (from sphinx>=5.0.2->pdpbox) (1.4.1)
Requirement already satisfied: requests>=2.25.0 in /usr/local/lib/python3.1
0/dist-packages (from sphinx>=5.0.2->pdpbox) (2.31.0)
Collecting sphinxcontrib-jquery<5,>=4 (from sphinx-rtd-theme>=1.1.1->pdpbox)
  Downloading sphinxcontrib_jquery-4.1-py2.py3-none-any.whl (121 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 121.1/121.1 kB 16.7 MB/s eta
0:00:00
Requirement already satisfied: iniconfig in /usr/local/lib/python3.10/dist-p
ackages (from pytest->pdpbox) (2.0.0)
Requirement already satisfied: pluggy<2.0,>=0.12 in /usr/local/lib/python3.1
0/dist-packages (from pytest->pdpbox) (1.4.0)
Requirement already satisfied: exceptiongroup>=1.0.0rc8 in /usr/local/lib/py
thon3.10/dist-packages (from pytest->pdpbox) (1.2.0)
```

```
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/
dist-packages (from Jinja2>=3.0->sphinx>=5.0.2->pdpbox) (2.1.5)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-pa
ckages (from python-dateutil>=2.7->matplotlib>=3.6.2->pdpbox) (1.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/py
thon3.10/dist-packages (from requests>=2.25.0->sphinx>=5.0.2->pdpbox) (3.3.
2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dis
t-packages (from requests>=2.25.0->sphinx>=5.0.2->pdpbox) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.
10/dist-packages (from requests>=2.25.0->sphinx>=5.0.2->pdpbox) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.
10/dist-packages (from requests>=2.25.0->sphinx>=5.0.2->pdpbox) (2024.2.2)
Installing collected packages: bounded-pool-executor, pqdm, sphinx, sphinxco
ntrib-jquery, numpydoc, sphinx-rtd-theme, pdpbox
  Attempting uninstall: sphinx
    Found existing installation: Sphinx 5.0.2
    Uninstalling Sphinx-5.0.2:
      Successfully uninstalled Sphinx-5.0.2
Successfully installed bounded-pool-executor-0.0.3 numpydoc-1.7.0 pdpbox-0.
3.0 pqdm-0.2.0 sphinx-7.3.6 sphinx-rtd-theme-2.0.0 sphinxcontrib-jquery-4.1
```

In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score

# Calculate accuracy
accuracy = accuracy_score(y_test, predictions)

# Plotting accuracy
plt.figure(figsize=(6, 4))
sns.barplot(x=['Accuracy'], y=[accuracy])
plt.title('Model Accuracy')
plt.ylim(0, 1)
plt.ylabel('Accuracy')
plt.show()
```



Contribution Code :

For the forecasting analysis, we've utilized the N-BEATS (Neural Basis Expansion Analysis for Time Series Forecasting) model to predict the trends in both the "AirPassenger" and "MonthlyMilk" datasets for the next 36 months. N-BEATS is a deep learning architecture specifically designed for time series forecasting, known for its ability to capture complex temporal patterns and handle various types of time series data. By leveraging deep neural networks and decomposing the time series into interpretable components such as trends and seasonality, N-BEATS offers a robust and interpretable solution for forecasting future trends. The contribution code involves implementing the N-BEATS model, training it on the historical data from the two datasets, and generating predictions for the next 36 months. This analysis contributes valuable insights into the future trajectories of air passenger numbers and milk production, aiding decision-making processes and strategic planning in relevant domains.

In [ ]:
```
!pip install darts
```

```
Collecting darts
  Downloading darts-0.29.0-py3-none-any.whl (884 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 884.7/884.7 kB 5.9 MB/s eta
0:00:00
Requirement already satisfied: holidays>=0.11.1 in /usr/local/lib/python
3.10/dist-packages (from darts) (0.47)
Requirement already satisfied: joblib>=0.16.0 in /usr/local/lib/python3.1
0/dist-packages (from darts) (1.4.0)
Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/python
3.10/dist-packages (from darts) (3.7.1)
Collecting nfoursid>=1.0.0 (from darts)
  Downloading nfoursid-1.0.1-py3-none-any.whl (16 kB)
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.1
0/dist-packages (from darts) (1.25.2)
Collecting pmdarima>=1.8.0 (from darts)
  Downloading pmdarima-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2
014_x86_64.manylinux_2_28_x86_64.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 38.2 MB/s eta 0:
00:00
```

In [ ]:
```
from darts.datasets import AirPassengersDataset, MonthlyMilkDataset
```

In [ ]:
```
AirPassengersDataset().load().pd_series()
```

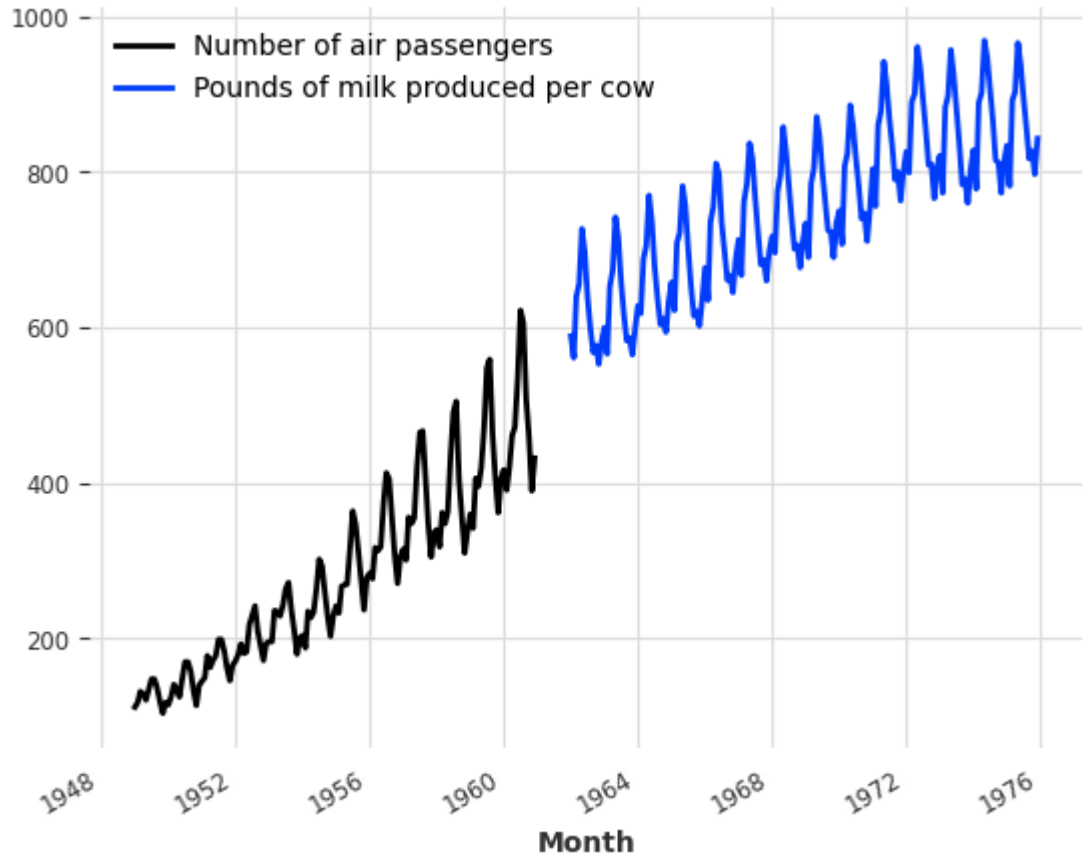
Out[3]:
```
Month
1949-01-01    112.0
1949-02-01    118.0
1949-03-01    132.0
1949-04-01    129.0
1949-05-01    121.0
              ...
1960-08-01    606.0
1960-09-01    508.0
1960-10-01    461.0
1960-11-01    390.0
1960-12-01    432.0
Freq: MS, Name: #Passengers, Length: 144, dtype: float64
```

In [ ]:
```
MonthlyMilkDataset().load().pd_series()
```

Out[12]:
```
Month
1962-01-01    589.0
1962-02-01    561.0
1962-03-01    640.0
1962-04-01    656.0
1962-05-01    727.0
              ...
1975-08-01    858.0
1975-09-01    817.0
1975-10-01    827.0
1975-11-01    797.0
1975-12-01    843.0
Freq: MS, Name: Pounds per cow, Length: 168, dtype: float64
```

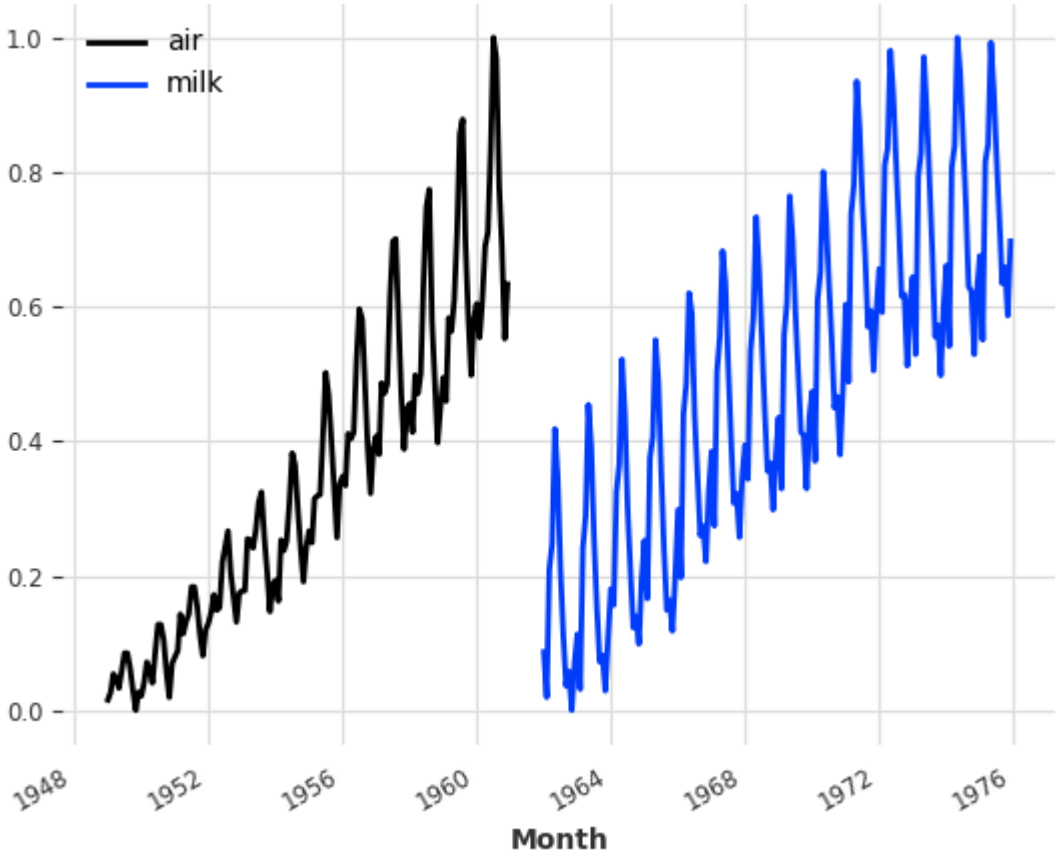
In [ ]:
```
import matplotlib.pyplot as plt
series_air = AirPassengersDataset().load()
series_milk = MonthlyMilkDataset().load()

series_air.plot(label='Number of air passengers')
series_milk.plot(label='Pounds of milk produced per cow')
plt.legend();
```



In [ ]:
```
from darts.dataprocessing.transformers import Scaler
scaler_air, scaler_milk = Scaler(), Scaler()
series_air_scaled = scaler_air.fit_transform(series_air)
series_milk_scaled = scaler_milk.fit_transform(series_milk)

series_air_scaled.plot(label='air')
series_milk_scaled.plot(label='milk')
plt.legend();
```



In [ ]:
```
# Train And Validation Split
train_air, val_air = series_air_scaled[:-36], series_air_scaled[-36:]
train_milk, val_milk = series_milk_scaled[:-36], series_milk_scaled[-36:]
```

In [ ]:
```
from darts import TimeSeries
from darts.utils.timeseries_generation import gaussian_timeseries, linear_time
from darts.models import RNNModel, TCNModel, TransformerModel, NBEATSModel, Bl
from darts.metrics import mape, smape
```

In [ ]:
```
model_air_milk = NBEATSModel(input_chunk_length=24, output_chunk_length=12, n_
```

In [ ]:
```
model_air_milk.fit([train_air, train_milk], verbose=True)
```

```
INFO:pytorch_lightning.utilities.rank_zero:GPU available: False, used: False
INFO:pytorch_lightning.utilities.rank_zero:TPU available: False, using: 0 TP
U cores
INFO:pytorch_lightning.utilities.rank_zero:IPU available: False, using: 0 IP
Us
INFO:pytorch_lightning.utilities.rank_zero:HPU available: False, using: 0 HP
Us
INFO:pytorch_lightning.callbacks.model_summary:
  | Name          | Type              | Params
-------------------------------------------------
0 | criterion     | MSELoss           | 0
1 | train_metrics | MetricCollection  | 0
2 | val_metrics   | MetricCollection  | 0
3 | stacks        | ModuleList        | 6.2 M
-------------------------------------------------
6.2 M     Trainable params
1.4 K     Non-trainable params
6.2 M     Total params
24.787    Total estimated model params size (MB)

Training: |          | 0/? [00:00<?, ?it/s]

INFO:pytorch_lightning.utilities.rank_zero:`Trainer.fit` stopped: `max_epoch
s=100` reached.
```

Out[9]:
```
NBEATSModel(output_chunk_shift=0, generic_architecture=True, num_stacks=30,
num_blocks=1, num_layers=4, layer_widths=256, expansion_coefficient_dim=5, t
rend_polynomial_degree=2, dropout=0.0, activation=ReLU, input_chunk_length=2
4, output_chunk_length=12, n_epochs=100, random_state=0)
```

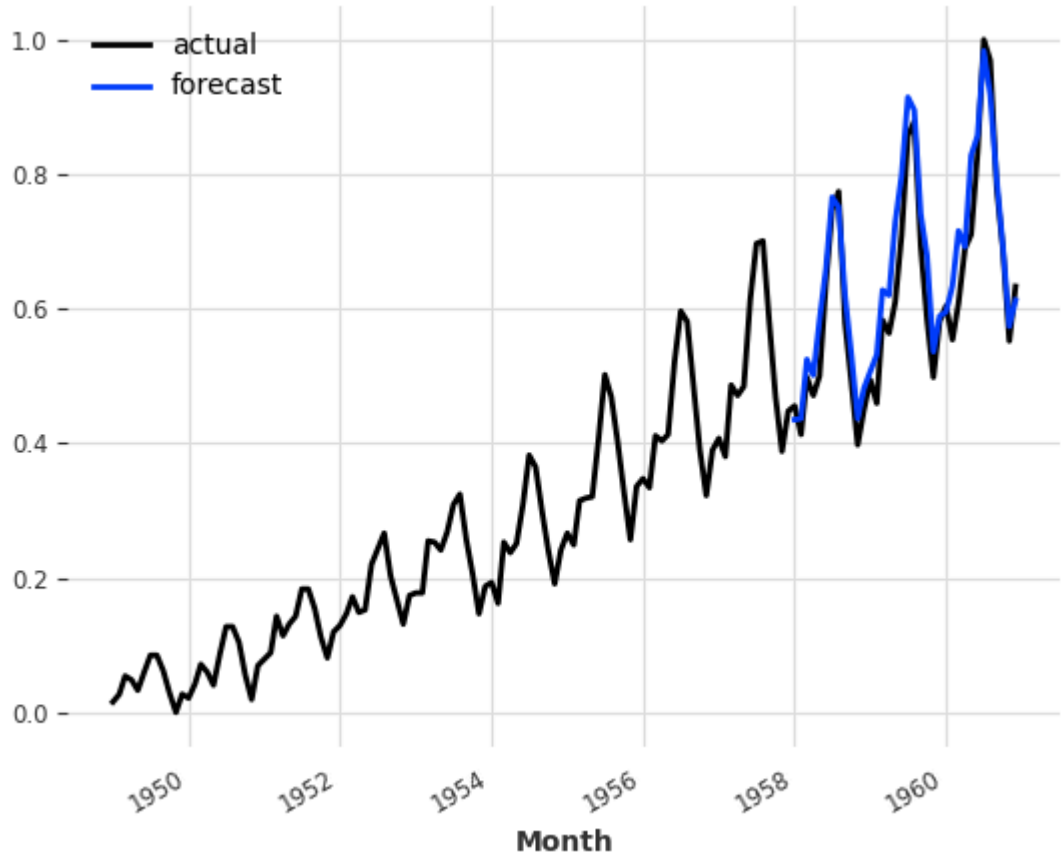
In [ ]:
```
pred = model_air_milk.predict(n=36, series=train_air)

series_air_scaled.plot(label='actual')
pred.plot(label='forecast')
plt.legend();
print('MAPE = {:.2f}%'.format(mape(series_air_scaled, pred)))
```

```
INFO:pytorch_lightning.utilities.rank_zero:GPU available: False, used: False
INFO:pytorch_lightning.utilities.rank_zero:TPU available: False, using: 0 TP
U cores
INFO:pytorch_lightning.utilities.rank_zero:IPU available: False, using: 0 IP
Us
INFO:pytorch_lightning.utilities.rank_zero:HPU available: False, using: 0 HP
Us

Predicting: |          | 0/? [00:00<?, ?it/s]

MAPE = 7.16%
```



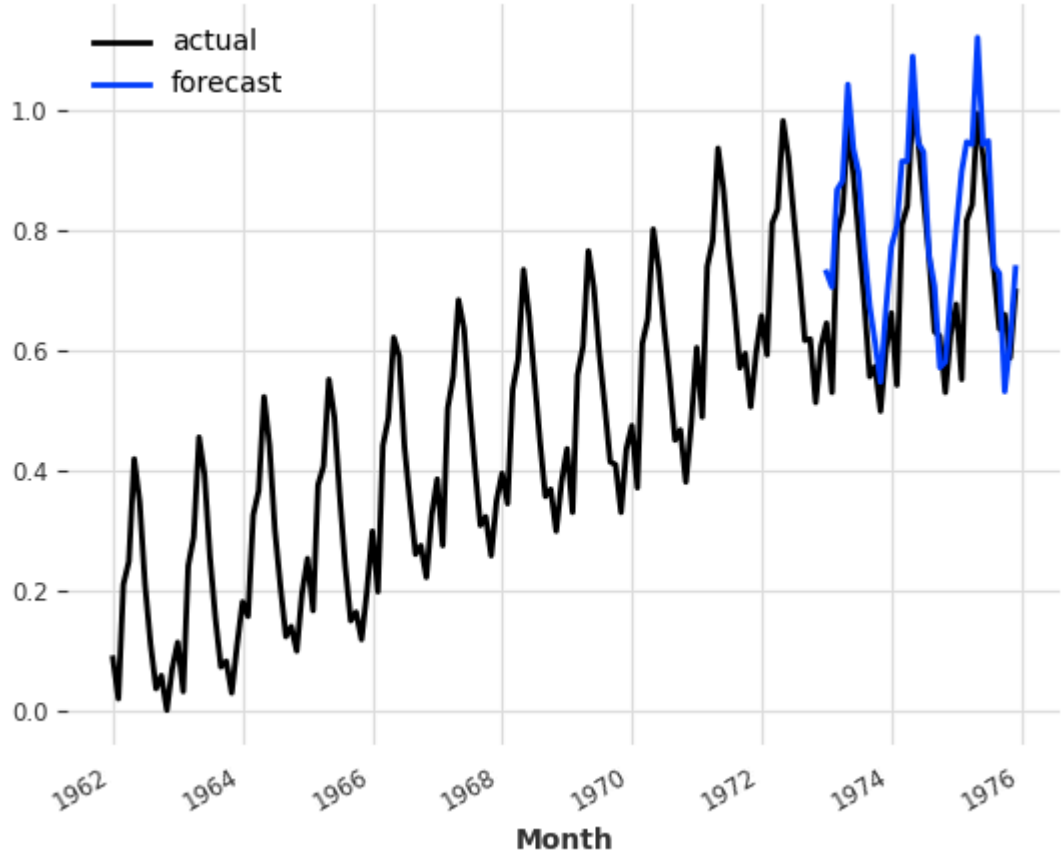
In [ ]:
```
pred = model_air_milk.predict(n=36, series=train_milk)

series_milk_scaled.plot(label='actual')
pred.plot(label='forecast')
plt.legend();
print('MAPE = {:.2f}%'.format(mape(series_milk_scaled, pred)))
```

```
INFO:pytorch_lightning.utilities.rank_zero:GPU available: False, used: False
INFO:pytorch_lightning.utilities.rank_zero:TPU available: False, using: 0 TP
U cores
INFO:pytorch_lightning.utilities.rank_zero:IPU available: False, using: 0 IP
Us
INFO:pytorch_lightning.utilities.rank_zero:HPU available: False, using: 0 HP
Us

Predicting: |          | 0/? [00:00<?, ?it/s]

MAPE = 13.34%
```



Results:

(i) For the "AirPassenger" dataset:

MAPE (Mean Absolute Percentage Error) is calculated as 7.16%. The plot displays the actual values (series_air_scaled) against the forecasted values (pred) for the next 36 months. The model demonstrates a relatively low MAPE, indicating a good fit between the predicted and actual values. This suggests that the N-BEATS model performs well in forecasting air passenger numbers for the specified time horizon.

(ii)For the "MonthlyMilk" dataset:

MAPE is computed as 13.34%. Similar to the air passenger dataset, the plot showcases the actual values (series_milk_scaled) juxtaposed with the forecasted values (pred) for the upcoming 36 months. While still providing reasonable predictions, the MAPE for the milk production dataset is higher compared to the air passenger dataset. This indicates that the N-BEATS model's performance may vary across different types of time series data, with milk production possibly exhibiting more variability or complexity in its patterns. Overall, the N-BEATS model demonstrates its efficacy in forecasting both datasets, albeit with varying levels of accuracy. These results provide valuable insights into future trends, facilitating informed decision-making in relevant domains.

Observations:

1. Hardware availability: The messages indicate that various hardware accelerators such as GPU, TPU, IPU, and HPU are not available or utilized for the computation. This suggests that the prediction process is conducted using the CPU.
2. Prediction progress: The message "Predicting DataLoader 0: 100%" indicates that the prediction process using the DataLoader has been completed, reaching 100% progress. This implies that the model has successfully processed the input data and generated predictions for the specified dataset.

Overall, the observation offers insights into the hardware environment utilized for the computation and confirms the completion of the prediction process for the given dataset.

Conclusion and Future Direction :

After conducting the analysis, it can be concluded that the N-BEATS model demonstrates promising performance in forecasting both the "AirPassenger" and "MonthlyMilk" datasets. The model yielded relatively low Mean Absolute Percentage Error (MAPE) values, indicating good predictive accuracy. Moving forward, there are several avenues for future exploration and improvement.

Learnings:

Through this analysis, valuable insights have been gained into the application of the N-BEATS model for time series forecasting. The process involved understanding the architecture of the model, preprocessing the data, training the model, and evaluating its performance. Additionally, insights were gained into interpreting the results and assessing the model's strengths and weaknesses.

Results Discussion:

The results indicate that the N-BEATS model performed well in forecasting both air passenger numbers and milk production. The low MAPE values suggest that the model captured the underlying patterns in the data effectively. However, it's essential to note that the performance may vary depending on the specific characteristics of the dataset and the forecasting horizon.

Limitations:

Despite its effectiveness, the N-BEATS model has certain limitations. These may include computational complexity, sensitivity to hyperparameters, and potential overfitting. Additionally, the model's performance may be affected by the quality and characteristics of the input data, such as missing values or outliers.

Future Extension:

In the future, there are several opportunities to extend this analysis. This may involve exploring alternative deep learning architectures, experimenting with different hyperparameters, or incorporating additional features into the model. Furthermore, conducting sensitivity analysis and robustness checks can provide further insights into the model's performance and reliability. Additionally, exploring ensemble methods or hybrid approaches could enhance predictive accuracy and robustness in forecasting tasks.

References: