In [1]:
```
import warnings; warnings.simplefilter('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
```

HW 3: Classification for time series forecasting

Let's read the air_passenger.csv file using read_csv function in pandas package. Make sure index is a date/time object with a monthly frequency. Print out first five rows of the series. Print out the frequency of the index.

In [2]:
```
air=pd.read_csv('air_passenger.csv',index_col='date',parse_dates=True)
#same file that you used for the previous HW.
air.index.freq = 'M'
print(air.head())
```

```
            passengers
date
1949-01-31         112
1949-02-28         118
1949-03-31         132
1949-04-30         129
1949-05-31         121
```

Let's create a variable to to see $X_t - X_{t-1}$.

In [3]:
```
air['diff_air']=air['passengers'].diff(1)
```

We will work on classification models to predict the up on down of this difference based on lagged variables. Let's create lagged variables and process the data to prep the analysis.

In [4]:
```
lags= [1,2,3,4,5,6,7,8,9,10,11,12]
for lag in lags:
    air[f'lag_{lag}'] = air['passengers'].shift(lag)
```

```python
air=air.dropna()

air['direction'] = 'up'
air.loc[(air['diff_air'] < 0) , 'direction'] = 'down'
air = air.drop(columns=['diff_air','passengers'])
```

In [5]:
```python
print(air.head(5))
```

```
            lag_1  lag_2  lag_3  lag_4  lag_5  lag_6  lag_7  lag_8  lag_9  \
date
1950-01-31  118.0  104.0  119.0  136.0  148.0  148.0  135.0  121.0  129.0
1950-02-28  115.0  118.0  104.0  119.0  136.0  148.0  148.0  135.0  121.0
1950-03-31  126.0  115.0  118.0  104.0  119.0  136.0  148.0  148.0  135.0
1950-04-30  141.0  126.0  115.0  118.0  104.0  119.0  136.0  148.0  148.0
1950-05-31  135.0  141.0  126.0  115.0  118.0  104.0  119.0  136.0  148.0

            lag_10  lag_11  lag_12 direction
date
1950-01-31   132.0   118.0   112.0      down
1950-02-28   129.0   132.0   118.0        up
1950-03-31   121.0   129.0   132.0        up
1950-04-30   135.0   121.0   129.0      down
1950-05-31   148.0   135.0   121.0      down
```

For this part, conduct the following:

Train/test split:

- Training set: observations in 1954,
- Test set: observations in 1955.

In [6]:
```python
# Train/test split by year: Training set (1954), Test set (1955)
train = air.loc['1954']
test = air.loc['1955']

# Print basic info about the splits
print("Training set (1954):", train.shape)
print(train.head())
```

```python
print("\nTest set (1955):", test.shape)
print(test.head())
```

```
Training set (1954): (12, 13)
            lag_1  lag_2  lag_3  lag_4  lag_5  lag_6  lag_7  lag_8  lag_9  \
date
1954-01-31  201.0  180.0  211.0  237.0  272.0  264.0  243.0  229.0  235.0
1954-02-28  204.0  201.0  180.0  211.0  237.0  272.0  264.0  243.0  229.0
1954-03-31  188.0  204.0  201.0  180.0  211.0  237.0  272.0  264.0  243.0
1954-04-30  235.0  188.0  204.0  201.0  180.0  211.0  237.0  272.0  264.0
1954-05-31  227.0  235.0  188.0  204.0  201.0  180.0  211.0  237.0  272.0

            lag_10  lag_11  lag_12 direction
date
1954-01-31   236.0   196.0   196.0        up
1954-02-28   235.0   236.0   196.0      down
1954-03-31   229.0   235.0   236.0        up
1954-04-30   243.0   229.0   235.0      down
1954-05-31   264.0   243.0   229.0        up

Test set (1955): (12, 13)
            lag_1  lag_2  lag_3  lag_4  lag_5  lag_6  lag_7  lag_8  lag_9  \
date
1955-01-31  229.0  203.0  229.0  259.0  293.0  302.0  264.0  234.0  227.0
1955-02-28  242.0  229.0  203.0  229.0  259.0  293.0  302.0  264.0  234.0
1955-03-31  233.0  242.0  229.0  203.0  229.0  259.0  293.0  302.0  264.0
1955-04-30  267.0  233.0  242.0  229.0  203.0  229.0  259.0  293.0  302.0
1955-05-31  269.0  267.0  233.0  242.0  229.0  203.0  229.0  259.0  293.0

            lag_10  lag_11  lag_12 direction
date
1955-01-31   235.0   188.0   204.0        up
1955-02-28   227.0   235.0   188.0      down
1955-03-31   234.0   227.0   235.0        up
1955-04-30   264.0   234.0   227.0        up
1955-05-31   302.0   264.0   234.0        up
```

Fit classification following classification models:

- logistic regression
- k-nn (test on multiple k values, recommend range from 1 to 12)

- decision tree
- random forest

In [7]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Define features and target for train and test sets
X_train = train[[f'lag_{i}' for i in range(1, 13)]]
y_train = train['direction']
X_test  = test[[f'lag_{i}' for i in range(1, 13)]]
y_test  = test['direction']

# Logistic Regression
lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)
print("Logistic Regression")
print("Accuracy:", accuracy_score(y_test, y_pred_lr))
print(classification_report(y_test, y_pred_lr))
print("-" * 50)

# k-NN (testing k from 1 to 12)
print("k-NN Results:")
for k in range(1, 13):
    knn_model = KNeighborsClassifier(n_neighbors=k)
    knn_model.fit(X_train, y_train)
    y_pred_knn = knn_model.predict(X_test)
    print(f"k = {k}: Accuracy = {accuracy_score(y_test, y_pred_knn):.4f}")
print("-" * 50)

# Decision Tree
tree_model = DecisionTreeClassifier(random_state=42)
tree_model.fit(X_train, y_train)
y_pred_tree = tree_model.predict(X_test)
print("Decision Tree")
print("Accuracy:", accuracy_score(y_test, y_pred_tree))
print(classification_report(y_test, y_pred_tree))
```

```
print("-" * 50)

# Random Forest
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
print("Random Forest")
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))
```

```
Logistic Regression
Accuracy: 0.75
              precision    recall  f1-score   support

        down       0.62      1.00      0.77         5
          up       1.00      0.57      0.73         7

    accuracy                           0.75        12
   macro avg       0.81      0.79      0.75        12
weighted avg       0.84      0.75      0.74        12

--------------------------------------------------
k-NN Results:
k = 1: Accuracy = 0.9167
k = 2: Accuracy = 0.6667
k = 3: Accuracy = 0.6667
k = 4: Accuracy = 0.5000
k = 5: Accuracy = 0.7500
k = 6: Accuracy = 0.6667
k = 7: Accuracy = 0.6667
k = 8: Accuracy = 0.6667
k = 9: Accuracy = 0.5000
k = 10: Accuracy = 0.5833
k = 11: Accuracy = 0.4167
k = 12: Accuracy = 0.4167
--------------------------------------------------
Decision Tree
Accuracy: 0.5833333333333334
              precision    recall  f1-score   support

        down       0.50      0.80      0.62         5
          up       0.75      0.43      0.55         7

    accuracy                           0.58        12
   macro avg       0.62      0.61      0.58        12
weighted avg       0.65      0.58      0.57        12

--------------------------------------------------
Random Forest
Accuracy: 0.6666666666666666
              precision    recall  f1-score   support
```

```
        down       0.57      0.80      0.67         5
          up       0.80      0.57      0.67         7

    accuracy                           0.67        12
   macro avg       0.69      0.69      0.67        12
weighted avg       0.70      0.67      0.67        12
```

And report the following of each models

- Accuracy
- Precision
- Recall

In [8]:

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score

# Logistic Regression
lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)
acc_lr = accuracy_score(y_test, y_pred_lr)
prec_lr = precision_score(y_test, y_pred_lr, pos_label='up')
rec_lr = recall_score(y_test, y_pred_lr, pos_label='up')
print("Logistic Regression")
print("Accuracy:", acc_lr)
print("Precision:", prec_lr)
print("Recall:", rec_lr)
print("-"*50)

# k-NN (testing k from 1 to 12)
print("k-NN Results:")
for k in range(1, 13):
    knn_model = KNeighborsClassifier(n_neighbors=k)
    knn_model.fit(X_train, y_train)
    y_pred_knn = knn_model.predict(X_test)
    acc_knn = accuracy_score(y_test, y_pred_knn)
    prec_knn = precision_score(y_test, y_pred_knn, pos_label='up')
    rec_knn = recall_score(y_test, y_pred_knn, pos_label='up')
    print(f"k = {k}: Accuracy = {acc_knn:.4f}, Precision = {prec_knn:.4f}, Recall = {rec_knn:.4f}")
```

```python
print("-"*50)

# Decision Tree
tree_model = DecisionTreeClassifier(random_state=42)
tree_model.fit(X_train, y_train)
y_pred_tree = tree_model.predict(X_test)
acc_tree = accuracy_score(y_test, y_pred_tree)
prec_tree = precision_score(y_test, y_pred_tree, pos_label='up')
rec_tree = recall_score(y_test, y_pred_tree, pos_label='up')
print("Decision Tree")
print("Accuracy:", acc_tree)
print("Precision:", prec_tree)
print("Recall:", rec_tree)
print("-"*50)

# Random Forest
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
acc_rf = accuracy_score(y_test, y_pred_rf)
prec_rf = precision_score(y_test, y_pred_rf, pos_label='up')
rec_rf = recall_score(y_test, y_pred_rf, pos_label='up')
print("Random Forest")
print("Accuracy:", acc_rf)
print("Precision:", prec_rf)
print("Recall:", rec_rf)
```

```
Logistic Regression
Accuracy: 0.75
Precision: 1.0
Recall: 0.5714285714285714
--------------------------------------------------
k-NN Results:
k = 1: Accuracy = 0.9167, Precision = 1.0000, Recall = 0.8571
k = 2: Accuracy = 0.6667, Precision = 1.0000, Recall = 0.4286
k = 3: Accuracy = 0.6667, Precision = 0.8000, Recall = 0.5714
k = 4: Accuracy = 0.5000, Precision = 0.6667, Recall = 0.2857
k = 5: Accuracy = 0.7500, Precision = 0.8333, Recall = 0.7143
k = 6: Accuracy = 0.6667, Precision = 1.0000, Recall = 0.4286
k = 7: Accuracy = 0.6667, Precision = 0.8000, Recall = 0.5714
k = 8: Accuracy = 0.6667, Precision = 1.0000, Recall = 0.4286
k = 9: Accuracy = 0.5000, Precision = 0.6000, Recall = 0.4286
k = 10: Accuracy = 0.5833, Precision = 1.0000, Recall = 0.2857
k = 11: Accuracy = 0.4167, Precision = 0.5000, Recall = 0.2857
k = 12: Accuracy = 0.4167, Precision = 0.0000, Recall = 0.0000
--------------------------------------------------
Decision Tree
Accuracy: 0.5833333333333334
Precision: 0.75
Recall: 0.42857142857142855
--------------------------------------------------
Random Forest
Accuracy: 0.6666666666666666
Precision: 0.8
Recall: 0.5714285714285714
```