Final year Project

Accuracy of models :

Code :

```
import pandas as pd
import numpy as np
import lightgbm as lgb
from sklearn.metrics import accuracy_score

elapsed_times = []
train_data_path = 'Trainset.csv'
train_df = pd.read_csv(train_data_path)
test_data_path = 'Testset1.csv'
test_df = pd.read_csv(test_data_path)

start_time = time.time()

X_train = train_df.drop('label', axis=1)
y_train = train_df['label']
X_test = test_df.drop('label', axis=1)
y_test = test_df['label']

clf = lgb.LGBMClassifier()
clf.fit(X_train, y_train)
predictions = clf.predict(X_test)

print("Accuracy: " + str(accuracy_score(y_test, predictions)))
```

Output ---Testset1:

```
Accuracy: 0.999675
```

Output ---Testset2:

```
Accuracy: 0.937175
```

1. KS test

code :

```
import pandas as pd
from scipy import stats

train_data = pd.read_csv('Trainset.csv')
test_data = pd.read_csv('Testset2.csv')

columns_to_test = train_data.columns

for column in columns_to_test:
    train_values = train_data[column]
    test_values = test_data[column]

    ks_stat, p_value = stats.ks_2samp(train_values, test_values)

    print(f"Column: {column}")
    print(f"KS Statistic: {ks_stat}")
    print(f"P-Value: {p_value}")



    if p_value < 0.05:
        print(f"Drift detected in column {column}!")
    else:
        print(f"No drift detected in column {column}.")
    print()
```

Output ---Testset1:

```
Column: label
KS Statistic: 0.0
P-Value: 1.0
No drift detected in column label.

Column: orig_ip_bytes
KS Statistic: 0.0005124999999999991
P-Value: 1.0
No drift detected in column orig_ip_bytes.

Column: orig_pkts
KS Statistic: 0.0005000000000000004
P-Value: 1.0
No drift detected in column orig_pkts.
```

Output ---Testset2:

```
Column: label
KS Statistic: 0.0
P-Value: 1.0
No drift detected in column label.

Column: orig_ip_bytes
KS Statistic: 0.06053749999999991
P-Value: 7.120070401448669e-128
Drift detected in column orig_ip_bytes!

Column: orig_pkts
KS Statistic: 0.0005000000000000004
P-Value: 1.0
No drift detected in column orig_pkts.
```

2.CUSUM test

Code:

```python
import pandas as pd
import numpy as np

train_df = pd.read_csv('Trainset.csv')
test_df = pd.read_csv('Testset2.csv')
```

```
column_to_monitor = 'orig_ip_bytes'

train_mean = train_df[column_to_monitor].mean()
train_std = train_df[column_to_monitor].std()

cusum = np.cumsum((test_df[column_to_monitor] - train_mean) / train_std)

drift_index = np.argmax(cusum)

drift_indices = np.where(cusum > 5)[0]
print(f"Drifts detected at indices {drift_indices}!")

if np.max(cusum) > 5:
    print("Drift detected!")
else:
    print("No drift detected!")
```

Output ---Testset1:

```
Drifts detected at indices [50019 50020 50021 ... 79997 79998 79999]!
```

Output ---Testset2:

```
Drifts detected at indices [50019 50020 50021 ... 79997 79998 79999]!
label                   3
orig_ip_bytes    139377944
orig_pkts          4806136
Name: 74748, dtype: int64
Drift detected!
PS C:\Users\HP\Desktop\final>
```

3. Logistic Regression

Code :

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

train_df = pd.read_csv('Trainset.csv')
test_df = pd.read_csv('Testset2.csv')
```

```
feature_cols = train_df.columns
target_col = 'orig_ip_bytes'

clf = LogisticRegression()
clf.fit(train_df[feature_cols], train_df[target_col])

y_pred_train = clf.predict(train_df[feature_cols])
train_accuracy = accuracy_score(train_df[target_col], y_pred_train)
print(f"Training accuracy: {train_accuracy:.3f}")

y_pred_test = clf.predict(test_df[feature_cols])
test_accuracy = accuracy_score(test_df[target_col], y_pred_test)
print(f"Test accuracy: {test_accuracy:.3f}")

if test_accuracy < train_accuracy * 0.8:
    print("Drift detected!")

    drift_indices = []
    for i, (y_pred, y_true) in enumerate(zip(y_pred_test, test_df[target_col])):
        if y_pred != y_true:
            drift_indices.append(i)
    print(f"Drift indices: {drift_indices}")
else:
    print("No drift detected.")
```

Output ---Testset1:

```
Training accuracy: 0.970
Test accuracy: 0.969
```

Output ---Testset2:

```
Training accuracy: 0.970
Test accuracy: 0.913
```

4.AdWin

Code:

```
import pandas as pd
from river import drift

train_df = pd.read_csv('Trainset.csv')
test_df = pd.read_csv('Testset1.csv')

columns_to_monitor = train_df.columns

adwin_objects = []

for column in columns_to_monitor:
    adwin_objects.append(drift.ADWIN(delta=0.002))

drifts = []

for i, row in test_df.iterrows():

    drift_detected = False


    for j, column in enumerate(columns_to_monitor):

        adwin_objects[j].update(row[column])


        if adwin_objects[j].drift_detected:
            drift_detected = True
            break


    if drift_detected:
        drifts.append((i, row[columns_to_monitor]))

        for adwin in adwin_objects:
            adwin_objects = [drift.ADWIN(delta=0.002) for _ in
columns_to_monitor]

for drift in drifts:
    print('Drift detected at index {}: \n{}'.format(drift[0], drift[1]))
```

Output ---Testset1:

```
Drift detected at index 40031:
label             3.0
orig_ip_bytes    80.0
orig_pkts         2.0
Name: 40031, dtype: float64
```

Output ---Testset2:

```
Drift detected at index 40031:
label             3
orig_ip_bytes    80
orig_pkts         2
Name: 40031, dtype: int64
```

5.KSWin

Code:

```
import pandas as pd
from river.drift import KSWIN

train_data = pd.read_csv('Trainset.csv')
test_data = pd.read_csv('Testset2.csv')

def detect_drift(train_data, test_data, feature_column):
    kswin = KSWIN()
    drift_indices = []

    for value in train_data[feature_column]:
        kswin.update(value)

        for index, value in enumerate(test_data[feature_column]):
        kswin.update(value)
        if kswin.drift_detected:
            drift_indices.append(index)
            print(f"Drift detected at index {index} in feature {feature_column}")

    if not drift_indices:
        print(f"No drift detected in feature {feature_column}")
```

```
    else:
        print(f"Drift detected at indices {drift_indices} in feature
{feature_column}")

for feature in train_data.columns:
    if feature != 'orig_ip_bytes':
        print(f"Checking drift in feature: {feature}")
        detect_drift(train_data, test_data, feature)
```

Output ---Testset1:

```
Checking drift in feature: label
Drift detected at index 13 in feature label
Drift detected at index 40013 in feature label
Drift detected at indices [13, 40013] in feature label
Checking drift in feature: orig_pkts
Drift detected at index 15 in feature orig_pkts
Drift detected at index 40012 in feature orig_pkts
Drift detected at indices [15, 40012] in feature orig_pkts
```

Output ---Testset2:

```
Checking drift in feature: label
Drift detected at index 13 in feature label
Drift detected at index 40013 in feature label
Drift detected at indices [13, 40013] in feature label
Checking drift in feature: orig_pkts
Drift detected at index 15 in feature orig_pkts
Drift detected at index 40012 in feature orig_pkts
Drift detected at indices [15, 40012] in feature orig_pkts
```

5.Page Hinkley

Code:

```python
import pandas as pd
import numpy as np

def page_hinkley_test(data, threshold=50, lambda_=50, alpha=1 - 1e-4):
    mean = 0
    sum_values = 0
    m_max = -np.inf
    change_point = None

    for i, x in enumerate(data):
        sum_values += x - mean - lambda_
        mean = (1 / (i + 1)) * ((i * mean) + x)

        m_t = sum_values / (i + 1)

        if m_t > m_max:
            m_max = m_t
        elif m_max - m_t > threshold:
            change_point = i
            break

    return change_point

train_data = pd.read_csv("Trainset.csv")
test_data = pd.read_csv("Testset2.csv")
```

```
columns_to_check = ['orig_ip_bytes', 'orig_pkts']

drift_results = {}
for col in columns_to_check:
    if test_data[col].dtype in (np.float64, np.int64):
        drift_index = page_hinkley_test(test_data[col].values)
        if drift_index is not None:
            drift_results[col] = f"Drift detected at index {drift_index}"
        else:
            drift_results[col] = "No drift detected"

print(drift_results)
```

Output ---Testset1:

```
{'orig_ip_bytes': 'Drift detected at index 1', 'orig_pkts': 'Drift detected at index 52199'}
```

Output ---Testset2:

```
{'orig_ip_bytes': 'Drift detected at index 1', 'orig_pkts': 'Drift detected at index 52199'}
```

S.no	Method	Testset1	Testset2	Conclusion
1.	Accuracy (Decision tree)	0.999675 No drift	0.937175 Drift	Since there is drift testset2, it has less accuracy
2.	KS test	No drift	Drift	Based on p-value, it is verified
3.	Cusum	No drift	Drift	Based on threshold value, it is verified
4.	Logistic Regression	Test Acc:0.969 No drift	Test Acc:0.913 Drift	Since there is drift testset2, it has less test accuracy
5.	Adwin	drift	drift	Windowing method not efficient for static dataset
6.	KsWin	drift	drift	Windowing method not efficient for static dataset.
7.	Page Hinkley	drift	Drift	Not efficent