```
%pip install numpy pandas river
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import kagglehub
```

```
path = kagglehub.dataset_download("mlg-ulb/creditcardfraud")

print("Path to dataset files:", path)
```

```
Downloading from https://www.kaggle.com/api/v1/datasets/download/mlg-ulb/creditcardfraud?dataset_version_number=3...
100%|██████████| 66.0M/66.0M [00:02<00:00, 23.1MB/s]Extracting files...

Path to dataset files: /root/.cache/kagglehub/datasets/mlg-ulb/creditcardfraud/versions/3
```

```
df = pd.read_csv(path+'/creditcard.csv')
df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	-0.221929	0.062723	0.
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	0.502292	0.219422	0.

5 rows × 31 columns

```
df.describe()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	...	2.848070e+05	2.84807(
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	-5.556467e-16	1.213481e-16	-2.406331e-15	...	1.654067e-16	-3.56859
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00	...	7.345240e-01	7.25701(
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01	...	-3.483038e+01	-1.093314
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01	...	-2.283949e-01	-5.42350
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5.142873e-02	...	-2.945017e-02	6.78194
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5.971390e-01	...	1.863772e-01	5.28553
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1.559499e+01	...	2.720284e+01	1.050309

8 rows × 31 columns

```
import river
from river.linear_model import LogisticRegression
from river.forest import ARFClassifier
from river.linear_model import PAClassifier
from river.ensemble import AdaBoostClassifier, ADWINBoostingClassifier
from river.tree import HoeffdingTreeClassifier
from river.imblearn import HardSamplingClassifier, RandomSampler
from river.metrics import F1, MCC, ROCAUC
from river.preprocessing import StandardScaler
from river.stream import iter_pandas
```

Logistic Regression Model

```
# @title Logistic Regression Model
# Instantiate the Logistic Regression classifier
ns_logreg_model = LogisticRegression()

# Instantiate the metrics
ns_logreg_f1 = F1()
ns_logreg_mcc = MCC()
ns_logreg_roc = ROCAUC()

# Separate features (X) and target (y)
X = df.drop('Class', axis=1)
y = df['Class']

# Iterate through the DataFrame instance by instance
for i, (x, y_true) in enumerate(iter_pandas(X, y)):

    # Predict and update metrics
    y_pred_proba = ns_logreg_model.predict_proba_one(x)
    y_pred = ns_logreg_model.predict_one(x)

    ns_logreg_f1.update(y_true, y_pred)
    ns_logreg_mcc.update(y_true, y_pred)

    # Only update ROC AUC if the model provides probability estimates
    if y_pred_proba:
        ns_logreg_roc.update(y_true, y_pred_proba)

    # Learn from the true label
    ns_logreg_model.learn_one(x, y_true)

```

```
    # Print scores every 10000 rows
    if (i + 1) % 10000 == 0:
        print(f"Row {i+1} (learn_one): F1 Score = {ns_logreg_f1.get():.4f}, MCC Score = {ns_logreg_mcc.get():.4f}, ROC AUC Score = {ns_logreg_roc.get():.4f}")

# Print final scores
print("\nFinal Scores with Logistic Regression with no Scaling:")
print(f"Final F1 Score = {ns_logreg_f1.get():.4f}")
print(f"Final MCC Score = {ns_logreg_mcc.get():.4f}")
print(f"Final ROC AUC Score = {ns_logreg_roc.get():.4f}")
```

```
Row 10000 (learn_one): F1 Score = 0.0000, MCC Score = -0.0039, ROC AUC Score = 0.4978
Row 20000 (learn_one): F1 Score = 0.0116, MCC Score = 0.0074, ROC AUC Score = 0.5036
Row 30000 (learn_one): F1 Score = 0.0105, MCC Score = 0.0074, ROC AUC Score = 0.5037
Row 40000 (learn_one): F1 Score = 0.0095, MCC Score = 0.0069, ROC AUC Score = 0.5034
Row 50000 (learn_one): F1 Score = 0.0067, MCC Score = 0.0037, ROC AUC Score = 0.5018
Row 60000 (learn_one): F1 Score = 0.0061, MCC Score = 0.0034, ROC AUC Score = 0.5017
Row 70000 (learn_one): F1 Score = 0.0057, MCC Score = 0.0032, ROC AUC Score = 0.5016
Row 80000 (learn_one): F1 Score = 0.0051, MCC Score = 0.0026, ROC AUC Score = 0.5013
Row 90000 (learn_one): F1 Score = 0.0047, MCC Score = 0.0024, ROC AUC Score = 0.5012
Row 100000 (learn_one): F1 Score = 0.0045, MCC Score = 0.0022, ROC AUC Score = 0.5011
Row 110000 (learn_one): F1 Score = 0.0250, MCC Score = 0.0229, ROC AUC Score = 0.5115
Row 120000 (learn_one): F1 Score = 0.0242, MCC Score = 0.0222, ROC AUC Score = 0.5111
Row 130000 (learn_one): F1 Score = 0.0229, MCC Score = 0.0209, ROC AUC Score = 0.5105
Row 140000 (learn_one): F1 Score = 0.0226, MCC Score = 0.0208, ROC AUC Score = 0.5104
Row 150000 (learn_one): F1 Score = 0.0408, MCC Score = 0.0389, ROC AUC Score = 0.5195
Row 160000 (learn_one): F1 Score = 0.0784, MCC Score = 0.0764, ROC AUC Score = 0.5383
Row 170000 (learn_one): F1 Score = 0.0776, MCC Score = 0.0756, ROC AUC Score = 0.5379
Row 180000 (learn_one): F1 Score = 0.0767, MCC Score = 0.0748, ROC AUC Score = 0.5375
Row 190000 (learn_one): F1 Score = 0.0753, MCC Score = 0.0735, ROC AUC Score = 0.5368
Row 200000 (learn_one): F1 Score = 0.0725, MCC Score = 0.0707, ROC AUC Score = 0.5355
Row 210000 (learn_one): F1 Score = 0.0709, MCC Score = 0.0691, ROC AUC Score = 0.5346
Row 220000 (learn_one): F1 Score = 0.0688, MCC Score = 0.0671, ROC AUC Score = 0.5336
Row 230000 (learn_one): F1 Score = 0.0667, MCC Score = 0.0650, ROC AUC Score = 0.5325
Row 240000 (learn_one): F1 Score = 0.0662, MCC Score = 0.0645, ROC AUC Score = 0.5323
Row 250000 (learn_one): F1 Score = 0.0632, MCC Score = 0.0615, ROC AUC Score = 0.5308
Row 260000 (learn_one): F1 Score = 0.0614, MCC Score = 0.0597, ROC AUC Score = 0.5299
Row 270000 (learn_one): F1 Score = 0.0602, MCC Score = 0.0585, ROC AUC Score = 0.5293
Row 280000 (learn_one): F1 Score = 0.0593, MCC Score = 0.0577, ROC AUC Score = 0.5289

Final Scores with Logistic Regression with no Scaling:
Final F1 Score = 0.0588
Final MCC Score = 0.0572
Final ROC AUC Score = 0.5286
```

Online Learning Models with Scaling

Scaled Logistic Regression Model

```
# @title Scaled Logistic Regression Model

# Instantiate the Logistic Regression classifier
logreg_model = StandardScaler() | LogisticRegression()

# Instantiate the metrics
logreg_f1 = F1()
logreg_mcc = MCC()
logreg_roc = ROCAUC()

# Separate features (X) and target (y)
X = df.drop('Class', axis=1)
y = df['Class']

# Iterate through the DataFrame instance by instance
for i, (x, y_true) in enumerate(iter_pandas(X, y)):

    # Predict and update metrics
    y_pred_proba = logreg_model.predict_proba_one(x)
    y_pred = logreg_model.predict_one(x)

    logreg_f1.update(y_true, y_pred)
    logreg_mcc.update(y_true, y_pred)

    # Only update ROC AUC if the model provides probability estimates
    if y_pred_proba:
         logreg_roc.update(y_true, y_pred_proba)

    # Learn from the true label
    logreg_model.learn_one(x, y_true)

    # Print scores every 10000 rows
    if (i + 1) % 10000 == 0:
        print(f"Row {i+1} (learn_one): F1 Score = {logreg_f1.get():.4f}, MCC Score = {logreg_mcc.get():.4f}, ROC AUC Score = {logreg_roc.get():.4f}")

# Print final scores
print("\nFinal Scores with Logistic Regression (learn_one):")
print(f"Final F1 Score = {logreg_f1.get():.4f}")
print(f"Final MCC Score = {logreg_mcc.get():.4f}")
print(f"Final ROC AUC Score = {logreg_roc.get():.4f}")
```

```
Row 10000 (learn_one): F1 Score = 0.8378, MCC Score = 0.8375, ROC AUC Score = 0.9457
Row 20000 (learn_one): F1 Score = 0.7229, MCC Score = 0.7219, ROC AUC Score = 0.8922
Row 30000 (learn_one): F1 Score = 0.7006, MCC Score = 0.7010, ROC AUC Score = 0.8708
Row 40000 (learn_one): F1 Score = 0.7179, MCC Score = 0.7189, ROC AUC Score = 0.8739
Row 50000 (learn_one): F1 Score = 0.7698, MCC Score = 0.7708, ROC AUC Score = 0.9046
Row 60000 (learn_one): F1 Score = 0.7641, MCC Score = 0.7662, ROC AUC Score = 0.8919
Row 70000 (learn_one): F1 Score = 0.7640, MCC Score = 0.7664, ROC AUC Score = 0.8908
Row 80000 (learn_one): F1 Score = 0.7744, MCC Score = 0.7772, ROC AUC Score = 0.8923
Row 90000 (learn_one): F1 Score = 0.7855, MCC Score = 0.7883, ROC AUC Score = 0.8977
Row 100000 (learn_one): F1 Score = 0.7833, MCC Score = 0.7867, ROC AUC Score = 0.8920
Row 110000 (learn_one): F1 Score = 0.7852, MCC Score = 0.7891, ROC AUC Score = 0.8887
Row 120000 (learn_one): F1 Score = 0.7795, MCC Score = 0.7831, ROC AUC Score = 0.8863
```

```
Row 130000 (learn_one): F1 Score = 0.7702, MCC Score = 0.7746, ROC AUC Score = 0.8847
Row 140000 (learn_one): F1 Score = 0.7705, MCC Score = 0.7750, ROC AUC Score = 0.8841
Row 150000 (learn_one): F1 Score = 0.7698, MCC Score = 0.7738, ROC AUC Score = 0.8853
Row 160000 (learn_one): F1 Score = 0.7938, MCC Score = 0.7971, ROC AUC Score = 0.8972
Row 170000 (learn_one): F1 Score = 0.7903, MCC Score = 0.7934, ROC AUC Score = 0.8969
Row 180000 (learn_one): F1 Score = 0.7910, MCC Score = 0.7942, ROC AUC Score = 0.8967
Row 190000 (learn_one): F1 Score = 0.7906, MCC Score = 0.7937, ROC AUC Score = 0.8973
Row 200000 (learn_one): F1 Score = 0.7846, MCC Score = 0.7881, ROC AUC Score = 0.8933
Row 210000 (learn_one): F1 Score = 0.7861, MCC Score = 0.7893, ROC AUC Score = 0.8944
Row 220000 (learn_one): F1 Score = 0.7838, MCC Score = 0.7872, ROC AUC Score = 0.8926
Row 230000 (learn_one): F1 Score = 0.7880, MCC Score = 0.7913, ROC AUC Score = 0.8936
Row 240000 (learn_one): F1 Score = 0.7899, MCC Score = 0.7935, ROC AUC Score = 0.8934
Row 250000 (learn_one): F1 Score = 0.7938, MCC Score = 0.7973, ROC AUC Score = 0.8939
Row 260000 (learn_one): F1 Score = 0.7944, MCC Score = 0.7981, ROC AUC Score = 0.8926
Row 270000 (learn_one): F1 Score = 0.7968, MCC Score = 0.8004, ROC AUC Score = 0.8938
Row 280000 (learn_one): F1 Score = 0.7928, MCC Score = 0.7964, ROC AUC Score = 0.8912

Final Scores with Logistic Regression (learn_one):
Final F1 Score = 0.7933
Final MCC Score = 0.7969
Final ROC AUC Score = 0.8911
```

Adaptive Random Forest Model

```python
# @title Adaptive Random Forest Model

# Initialize the model: a StandardScaler followed by an Adaptive Random Forest Classifier
ARF_model = StandardScaler() | ARFClassifier(n_models=10, seed=42)

# Instantiate the metrics
ARF_f1 = F1()
ARF_mcc = MCC()
ARF_roc = ROCAUC()

# Separate features (X) and target (y)
X = df.drop('Class', axis=1)
y = df['Class']

# Iterate through the DataFrame in chunks
for i, (x, y_true) in enumerate(iter_pandas(X, y)):

    # Predict and update metrics
    y_pred_proba = ARF_model.predict_proba_one(x)
    y_pred = ARF_model.predict_one(x)

    ARF_f1.update(y_true, y_pred)
    ARF_mcc.update(y_true, y_pred)

    # Only update ROC AUC if the model provides probability estimates
    if y_pred_proba:
         ARF_roc.update(y_true, y_pred_proba)

    # Learn from the true label
    ARF_model.learn_one(x, y_true)

    # Print scores every 10000 rows
    if (i + 1) % 10000 == 0:
        print(f"Row {i+1}: F1 Score = {ARF_f1.get():.4f}, MCC Score = {ARF_mcc.get():.4f}, ROC AUC Score = {ARF_roc.get():.4f}")


# Print final scores
print("Final Scores after processing all data with ARFClassifier model:")
print(f"Final F1 Score = {ARF_f1.get():.4f}")
print(f"Final MCC Score = {ARF_mcc.get():.4f}")
print(f"Final ROC AUC Score = {ARF_roc.get():.4f}")
```

```
Row 10000: F1 Score = 0.5926, MCC Score = 0.6482, ROC AUC Score = 0.7497
Row 20000: F1 Score = 0.6519, MCC Score = 0.6739, ROC AUC Score = 0.8288
Row 30000: F1 Score = 0.6111, MCC Score = 0.6411, ROC AUC Score = 0.8134
Row 40000: F1 Score = 0.5897, MCC Score = 0.6249, ROC AUC Score = 0.8218
Row 50000: F1 Score = 0.6552, MCC Score = 0.6810, ROC AUC Score = 0.8578
Row 60000: F1 Score = 0.6154, MCC Score = 0.6489, ROC AUC Score = 0.8311
Row 70000: F1 Score = 0.5869, MCC Score = 0.6263, ROC AUC Score = 0.8226
Row 80000: F1 Score = 0.5532, MCC Score = 0.6002, ROC AUC Score = 0.8161
Row 90000: F1 Score = 0.5253, MCC Score = 0.5785, ROC AUC Score = 0.8126
Row 100000: F1 Score = 0.5049, MCC Score = 0.5627, ROC AUC Score = 0.8003
Row 110000: F1 Score = 0.4800, MCC Score = 0.5436, ROC AUC Score = 0.7990
Row 120000: F1 Score = 0.4685, MCC Score = 0.5347, ROC AUC Score = 0.7934
Row 130000: F1 Score = 0.4496, MCC Score = 0.5202, ROC AUC Score = 0.7872
Row 140000: F1 Score = 0.4457, MCC Score = 0.5172, ROC AUC Score = 0.7840
Row 150000: F1 Score = 0.4444, MCC Score = 0.5178, ROC AUC Score = 0.7866
Row 160000: F1 Score = 0.5203, MCC Score = 0.5812, ROC AUC Score = 0.8131
Row 170000: F1 Score = 0.5161, MCC Score = 0.5780, ROC AUC Score = 0.8124
Row 180000: F1 Score = 0.5120, MCC Score = 0.5749, ROC AUC Score = 0.8103
Row 190000: F1 Score = 0.5049, MCC Score = 0.5694, ROC AUC Score = 0.8058
Row 200000: F1 Score = 0.4914, MCC Score = 0.5590, ROC AUC Score = 0.7986
Row 210000: F1 Score = 0.4830, MCC Score = 0.5526, ROC AUC Score = 0.7956
Row 220000: F1 Score = 0.4723, MCC Score = 0.5443, ROC AUC Score = 0.7955
Row 230000: F1 Score = 0.4640, MCC Score = 0.5380, ROC AUC Score = 0.7935
Row 240000: F1 Score = 0.4495, MCC Score = 0.5268, ROC AUC Score = 0.7905
Row 250000: F1 Score = 0.4336, MCC Score = 0.5146, ROC AUC Score = 0.7881
Row 260000: F1 Score = 0.4243, MCC Score = 0.5074, ROC AUC Score = 0.7876
Row 270000: F1 Score = 0.4175, MCC Score = 0.5021, ROC AUC Score = 0.7879
Row 280000: F1 Score = 0.4128, MCC Score = 0.4985, ROC AUC Score = 0.7847
Final Scores after processing all data with ARFClassifier model:
Final F1 Score = 0.4102
Final MCC Score = 0.4965
Final ROC AUC Score = 0.7834
```

PassiveAggresive Classifier Model

```python
# @title PassiveAggresive Classifier Model

```

```
# Instantiate the PassiveAggressiveClassifier
pa_model = StandardScaler() | PAClassifier(C=0.01, mode=1)

# Instantiate the metrics
f1_pa = F1()
mcc_pa = MCC()
roc_pa = ROCAUC()

# Separate features (X) and target (y)
X_pa_one = df.drop('Class', axis=1)
y_pa_one = df['Class']

# Iterate through the DataFrame instance by instance
for i, (x, y_true) in enumerate(iter_pandas(X_pa_one, y_pa_one)):

    # Predict and update metrics
    y_pred_proba = pa_model.predict_proba_one(x)
    y_pred = pa_model.predict_one(x)

    f1_pa.update(y_true, y_pred)
    mcc_pa.update(y_true, y_pred)

    # Only update ROC AUC if the model provides probability estimates
    if y_pred_proba:
         roc_pa.update(y_true, y_pred_proba)

    # Learn from the true label
    pa_model.learn_one(x, y_true)

    # Print scores every 10000 rows
    if (i + 1) % 10000 == 0:
        print(f"Row {i+1} (learn_one): F1 Score = {f1_pa.get():.4f}, MCC Score = {mcc_pa.get():.4f}, ROC AUC Score = {roc_pa.get():.4f}")

# Print final scores
print("\nFinal Scores with PassiveAggressiveClassifier (learn_one):")
print(f"Final F1 Score = {f1_pa.get():.4f}")
print(f"Final MCC Score = {mcc_pa.get():.4f}")
```

```
Row 10000 (learn_one): F1 Score = 0.8718, MCC Score = 0.8716, ROC AUC Score = 0.9834
Row 20000 (learn_one): F1 Score = 0.7561, MCC Score = 0.7556, ROC AUC Score = 0.9651
Row 30000 (learn_one): F1 Score = 0.7374, MCC Score = 0.7376, ROC AUC Score = 0.9537
Row 40000 (learn_one): F1 Score = 0.7538, MCC Score = 0.7539, ROC AUC Score = 0.9480
Row 50000 (learn_one): F1 Score = 0.8070, MCC Score = 0.8071, ROC AUC Score = 0.9616
Row 60000 (learn_one): F1 Score = 0.7871, MCC Score = 0.7876, ROC AUC Score = 0.9442
Row 70000 (learn_one): F1 Score = 0.7818, MCC Score = 0.7828, ROC AUC Score = 0.9420
Row 80000 (learn_one): F1 Score = 0.7913, MCC Score = 0.7924, ROC AUC Score = 0.9396
Row 90000 (learn_one): F1 Score = 0.8040, MCC Score = 0.8051, ROC AUC Score = 0.9451
Row 100000 (learn_one): F1 Score = 0.8010, MCC Score = 0.8025, ROC AUC Score = 0.9395
Row 110000 (learn_one): F1 Score = 0.8018, MCC Score = 0.8038, ROC AUC Score = 0.9317
Row 120000 (learn_one): F1 Score = 0.7930, MCC Score = 0.7950, ROC AUC Score = 0.9301
Row 130000 (learn_one): F1 Score = 0.7842, MCC Score = 0.7866, ROC AUC Score = 0.9337
Row 140000 (learn_one): F1 Score = 0.7844, MCC Score = 0.7868, ROC AUC Score = 0.9337
Row 150000 (learn_one): F1 Score = 0.7837, MCC Score = 0.7861, ROC AUC Score = 0.9336
Row 160000 (learn_one): F1 Score = 0.8060, MCC Score = 0.8076, ROC AUC Score = 0.9396
Row 170000 (learn_one): F1 Score = 0.8053, MCC Score = 0.8069, ROC AUC Score = 0.9420
Row 180000 (learn_one): F1 Score = 0.8047, MCC Score = 0.8063, ROC AUC Score = 0.9425
Row 190000 (learn_one): F1 Score = 0.8058, MCC Score = 0.8073, ROC AUC Score = 0.9444
Row 200000 (learn_one): F1 Score = 0.7994, MCC Score = 0.8012, ROC AUC Score = 0.9410
Row 210000 (learn_one): F1 Score = 0.8016, MCC Score = 0.8033, ROC AUC Score = 0.9417
Row 220000 (learn_one): F1 Score = 0.7979, MCC Score = 0.7997, ROC AUC Score = 0.9396
Row 230000 (learn_one): F1 Score = 0.7990, MCC Score = 0.8008, ROC AUC Score = 0.9405
Row 240000 (learn_one): F1 Score = 0.8005, MCC Score = 0.8025, ROC AUC Score = 0.9432
Row 250000 (learn_one): F1 Score = 0.8047, MCC Score = 0.8068, ROC AUC Score = 0.9425
Row 260000 (learn_one): F1 Score = 0.8050, MCC Score = 0.8073, ROC AUC Score = 0.9420
Row 270000 (learn_one): F1 Score = 0.8081, MCC Score = 0.8104, ROC AUC Score = 0.9425
Row 280000 (learn_one): F1 Score = 0.8004, MCC Score = 0.8025, ROC AUC Score = 0.9402

Final Scores with PassiveAggressiveClassifier (learn_one):
Final F1 Score = 0.8000
Final MCC Score = 0.8020
Final ROC AUC Score = 0.9398
```

AdaBoostClassifier with Hoeffding Tree Model

```
# @title AdaBoostClassifier with Hoeffding Tree Model

# Instantiate the AdaBoostClassifier with Hoeffding Tree as base model
ada_model = StandardScaler() | AdaBoostClassifier(model=HoeffdingTreeClassifier(), n_models=10,seed=42)

# Instantiate the metrics
f1_ada = F1()
mcc_ada = MCC()
roc_ada = ROCAUC()

# Separate features (X) and target (y)
X_ada = df.drop('Class', axis=1)
y_ada = df['Class']

# Iterate through the DataFrame instance by instance
for i, (x, y_true) in enumerate(iter_pandas(X_ada, y_ada)):

    # Predict and update metrics
    y_pred_proba = ada_model.predict_proba_one(x)
    y_pred = ada_model.predict_one(x)

    f1_ada.update(y_true, y_pred)
    mcc_ada.update(y_true, y_pred)

    if y_pred_proba:
      roc_ada.update(y_true, y_pred_proba)

    # Learn from the true label
    ada_model.learn_one(x, y_true)
```

```
    # Print scores every 10000 rows
    if (i + 1) % 10000 == 0:
        print(f"Row {i+1} (learn_one): F1 Score = {f1_ada.get():.4f}, MCC Score = {mcc_ada.get():.4f}, ROC AUC Score = {roc_ada.get():.4f}")

# Print final scores
print("\nFinal Scores with AdaBoostClassifier (learn_one):")
print(f"Final F1 Score = {f1_ada.get():.4f}")
print(f"Final MCC Score = {mcc_ada.get():.4f}")
```

```
Row 10000 (learn_one): F1 Score = 0.8919, MCC Score = 0.8918, ROC AUC Score = 0.9591
Row 20000 (learn_one): F1 Score = 0.8395, MCC Score = 0.8399, ROC AUC Score = 0.9710
Row 30000 (learn_one): F1 Score = 0.8090, MCC Score = 0.8097, ROC AUC Score = 0.9595
Row 40000 (learn_one): F1 Score = 0.8182, MCC Score = 0.8188, ROC AUC Score = 0.9568
Row 50000 (learn_one): F1 Score = 0.8429, MCC Score = 0.8438, ROC AUC Score = 0.9684
Row 60000 (learn_one): F1 Score = 0.8013, MCC Score = 0.8016, ROC AUC Score = 0.9546
Row 70000 (learn_one): F1 Score = 0.7906, MCC Score = 0.7905, ROC AUC Score = 0.9496
Row 80000 (learn_one): F1 Score = 0.8042, MCC Score = 0.8043, ROC AUC Score = 0.9535
Row 90000 (learn_one): F1 Score = 0.8039, MCC Score = 0.8040, ROC AUC Score = 0.9571
Row 100000 (learn_one): F1 Score = 0.7972, MCC Score = 0.7974, ROC AUC Score = 0.9534
Row 110000 (learn_one): F1 Score = 0.7922, MCC Score = 0.7922, ROC AUC Score = 0.9476
Row 120000 (learn_one): F1 Score = 0.7841, MCC Score = 0.7841, ROC AUC Score = 0.9416
Row 130000 (learn_one): F1 Score = 0.7753, MCC Score = 0.7755, ROC AUC Score = 0.9415
Row 140000 (learn_one): F1 Score = 0.7732, MCC Score = 0.7734, ROC AUC Score = 0.9395
Row 150000 (learn_one): F1 Score = 0.7690, MCC Score = 0.7690, ROC AUC Score = 0.9409
Row 160000 (learn_one): F1 Score = 0.7896, MCC Score = 0.7894, ROC AUC Score = 0.9467
Row 170000 (learn_one): F1 Score = 0.7875, MCC Score = 0.7872, ROC AUC Score = 0.9477
Row 180000 (learn_one): F1 Score = 0.7849, MCC Score = 0.7846, ROC AUC Score = 0.9481
Row 190000 (learn_one): F1 Score = 0.7824, MCC Score = 0.7821, ROC AUC Score = 0.9493
Row 200000 (learn_one): F1 Score = 0.7748, MCC Score = 0.7748, ROC AUC Score = 0.9476
Row 210000 (learn_one): F1 Score = 0.7706, MCC Score = 0.7706, ROC AUC Score = 0.9474
Row 220000 (learn_one): F1 Score = 0.7688, MCC Score = 0.7690, ROC AUC Score = 0.9442
Row 230000 (learn_one): F1 Score = 0.7683, MCC Score = 0.7684, ROC AUC Score = 0.9437
Row 240000 (learn_one): F1 Score = 0.7696, MCC Score = 0.7697, ROC AUC Score = 0.9442
Row 250000 (learn_one): F1 Score = 0.7694, MCC Score = 0.7698, ROC AUC Score = 0.9450
Row 260000 (learn_one): F1 Score = 0.7667, MCC Score = 0.7671, ROC AUC Score = 0.9455
Row 270000 (learn_one): F1 Score = 0.7682, MCC Score = 0.7687, ROC AUC Score = 0.9454
Row 280000 (learn_one): F1 Score = 0.7643, MCC Score = 0.7649, ROC AUC Score = 0.9445

Final Scores with AdaBoostClassifier (learn_one):
Final F1 Score = 0.7650
Final MCC Score = 0.7656
Final ROC AUC Score = 0.9438
```

AdaBoostClassifier with Scaled Hoeffding Tree

```
# @title AdaBoostClassifier with Scaled Hoeffding Tree

# Instantiate the AdaBoostClassifier with Hoeffding Tree as base model
ada_sc_model = AdaBoostClassifier(model=StandardScaler() |  HoeffdingTreeClassifier(), n_models=10,seed=42)

# Instantiate the metrics
f1_ada_sc = F1()
mcc_ada_sc = MCC()
roc_ada_sc = ROCAUC()

# Separate features (X) and target (y)
X_ada_sc = df.drop('Class', axis=1)
y_ada_sc = df['Class']

# Iterate through the DataFrame instance by instance
for i, (x, y_true) in enumerate(iter_pandas(X_ada_sc, y_ada_sc)):

    # Predict and update metrics
    y_pred_proba = ada_sc_model.predict_proba_one(x)
    y_pred = ada_sc_model.predict_one(x)

    f1_ada_sc.update(y_true, y_pred)
    mcc_ada_sc.update(y_true, y_pred)

    if y_pred_proba:
      roc_ada_sc.update(y_true, y_pred_proba)

    # Learn from the true label
    ada_sc_model.learn_one(x, y_true)

    # Print scores every 10000 rows
    if (i + 1) % 10000 == 0:
        print(f"Row {i+1} (learn_one): F1 Score = {f1_ada_sc.get():.4f}, MCC Score = {mcc_ada_sc.get():.4f}, ROC AUC Score = {roc_ada_sc.get():.4f}")

# Print final scores
print("Final Scores with AdaBoostClassifier (and scaled Hoeffding):")
print(f"Final F1 Score = {f1_ada_sc.get():.4f}")
print(f"Final MCC Score = {mcc_ada_sc.get():.4f}")
print(f"Final ROC AUC Score = {roc_ada_sc.get():.4f}")
```

```
Row 10000 (learn_one): F1 Score = 0.8571, MCC Score = 0.8598, ROC AUC Score = 0.9334
Row 20000 (learn_one): F1 Score = 0.8050, MCC Score = 0.8062, ROC AUC Score = 0.9353
Row 30000 (learn_one): F1 Score = 0.7791, MCC Score = 0.7818, ROC AUC Score = 0.9230
Row 40000 (learn_one): F1 Score = 0.7831, MCC Score = 0.7866, ROC AUC Score = 0.9306
Row 50000 (learn_one): F1 Score = 0.8291, MCC Score = 0.8311, ROC AUC Score = 0.9514
Row 60000 (learn_one): F1 Score = 0.8079, MCC Score = 0.8100, ROC AUC Score = 0.9405
Row 70000 (learn_one): F1 Score = 0.7975, MCC Score = 0.7993, ROC AUC Score = 0.9364
Row 80000 (learn_one): F1 Score = 0.8087, MCC Score = 0.8104, ROC AUC Score = 0.9365
Row 90000 (learn_one): F1 Score = 0.8152, MCC Score = 0.8167, ROC AUC Score = 0.9402
Row 100000 (learn_one): F1 Score = 0.8145, MCC Score = 0.8164, ROC AUC Score = 0.9325
Row 110000 (learn_one): F1 Score = 0.8172, MCC Score = 0.8194, ROC AUC Score = 0.9291
Row 120000 (learn_one): F1 Score = 0.8105, MCC Score = 0.8125, ROC AUC Score = 0.9244
Row 130000 (learn_one): F1 Score = 0.8025, MCC Score = 0.8051, ROC AUC Score = 0.9277
Row 140000 (learn_one): F1 Score = 0.7984, MCC Score = 0.8010, ROC AUC Score = 0.9264
Row 150000 (learn_one): F1 Score = 0.7911, MCC Score = 0.7935, ROC AUC Score = 0.9250
Row 160000 (learn_one): F1 Score = 0.8067, MCC Score = 0.8092, ROC AUC Score = 0.9314
```

```
Row 170000 (learn_one): F1 Score = 0.8066, MCC Score = 0.8088, ROC AUC Score = 0.9325
Row 180000 (learn_one): F1 Score = 0.8053, MCC Score = 0.8077, ROC AUC Score = 0.9319
Row 190000 (learn_one): F1 Score = 0.8023, MCC Score = 0.8050, ROC AUC Score = 0.9324
Row 200000 (learn_one): F1 Score = 0.7944, MCC Score = 0.7969, ROC AUC Score = 0.9314
Row 210000 (learn_one): F1 Score = 0.7945, MCC Score = 0.7967, ROC AUC Score = 0.9316
Row 220000 (learn_one): F1 Score = 0.7920, MCC Score = 0.7944, ROC AUC Score = 0.9293
Row 230000 (learn_one): F1 Score = 0.7908, MCC Score = 0.7927, ROC AUC Score = 0.9300
Row 240000 (learn_one): F1 Score = 0.7906, MCC Score = 0.7926, ROC AUC Score = 0.9313
Row 250000 (learn_one): F1 Score = 0.7925, MCC Score = 0.7947, ROC AUC Score = 0.9308
Row 260000 (learn_one): F1 Score = 0.7922, MCC Score = 0.7945, ROC AUC Score = 0.9312
Row 270000 (learn_one): F1 Score = 0.7955, MCC Score = 0.7978, ROC AUC Score = 0.9325
Row 280000 (learn_one): F1 Score = 0.7920, MCC Score = 0.7946, ROC AUC Score = 0.9321
Final Scores with AdaBoostClassifier (and scaled Hoeffding):
Final F1 Score = 0.7903
Final MCC Score = 0.7929
Final ROC AUC Score = 0.9316
```

Test Same models with Sampling Techniques

- In online Learning, HardSampling techniques is used to store data in the buffer from the dataset based on how 'hard' was it to predict the class and how the data affected the loss function.

AdaptiveRandomForest Classifier Model with Hardsampling

```python
# @title AdaptiveRandomForest Classifier Model with Hardsampling

# Instantiate the base ARFClassifier
base_model = ARFClassifier(n_models=10, seed=42)

# Instantiate HardSamplingClassifier, wrapping the base model

model_hs = HardSamplingClassifier(classifier=base_model, size=100, p=0.2, seed=42)


# Instantiate the hardOS metrics
f1_hs = F1()
mcc_hs = MCC()
roc_hs = ROCAUC()


# Separate features (X) and target (y)
X = df.drop('Class', axis=1)
y = df['Class']

# Iterate through the DataFrame
for i, (x, y_true) in enumerate(iter_pandas(X, y)):

    # Predict and update metrics for HardOS
    y_pred_proba = model_hs.predict_proba_one(x)
    y_pred = model_hs.predict_one(x)

    f1_hs.update(y_true, y_pred)
    mcc_hs.update(y_true, y_pred)


    if y_pred_proba: # Check if predict_proba_one returned probabilities
         roc_hs.update(y_true, y_pred_proba)


    # Learn from the true label using the HardSamplingClassifier
    model_hs.learn_one(x, y_true)

    # Print scores every 10000 rows
    if (i + 1) % 10000 == 0:
        print(f"\nRow {i+1} (HardSampling): F1 Score = {f1_hs.get():.4f}, MCC Score = {mcc_hs.get():.4f}, ROC AUC Score = {roc_hs.get():.4f}")


# # --- ARFClassifier with HardSamplingClassifier ---
print("--- ARFClassifier with HardSamplingClassifier ---")

# Print final scores
print("\nFinal Scores with HardSamplingClassifier:")
print(f"Final F1 Score = {f1_hs.get():.4f}")
print(f"Final MCC Score = {mcc_hs.get():.4f}")
print(f"Final ROC AUC Score = {roc_hs.get():.4f}")
```

```
Row 20000 (HardSampling): F1 Score = 0.7152, MCC Score = 0.7199, ROC AUC Score = 0.8818

Row 30000 (HardSampling): F1 Score = 0.6914, MCC Score = 0.6997, ROC AUC Score = 0.8718

Row 40000 (HardSampling): F1 Score = 0.7039, MCC Score = 0.7127, ROC AUC Score = 0.8793

Row 50000 (HardSampling): F1 Score = 0.7757, MCC Score = 0.7813, ROC AUC Score = 0.9118

Row 60000 (HardSampling): F1 Score = 0.7649, MCC Score = 0.7724, ROC AUC Score = 0.8983

Row 70000 (HardSampling): F1 Score = 0.7508, MCC Score = 0.7605, ROC AUC Score = 0.8965

Row 80000 (HardSampling): F1 Score = 0.7485, MCC Score = 0.7596, ROC AUC Score = 0.8999

Row 90000 (HardSampling): F1 Score = 0.7521, MCC Score = 0.7635, ROC AUC Score = 0.9070

Row 100000 (HardSampling): F1 Score = 0.7480, MCC Score = 0.7604, ROC AUC Score = 0.9030

Row 110000 (HardSampling): F1 Score = 0.7525, MCC Score = 0.7650, ROC AUC Score = 0.9010

Row 120000 (HardSampling): F1 Score = 0.7434, MCC Score = 0.7560, ROC AUC Score = 0.9022
```

```
Row 150000 (HardSampling): F1 Score = 0.7310, MCC Score = 0.7462, ROC AUC Score = 0.9020

Row 160000 (HardSampling): F1 Score = 0.7529, MCC Score = 0.7651, ROC AUC Score = 0.9122

Row 170000 (HardSampling): F1 Score = 0.7479, MCC Score = 0.7609, ROC AUC Score = 0.9131

Row 180000 (HardSampling): F1 Score = 0.7471, MCC Score = 0.7603, ROC AUC Score = 0.9127

Row 190000 (HardSampling): F1 Score = 0.7428, MCC Score = 0.7566, ROC AUC Score = 0.9117

Row 200000 (HardSampling): F1 Score = 0.7344, MCC Score = 0.7496, ROC AUC Score = 0.9083

Row 210000 (HardSampling): F1 Score = 0.7301, MCC Score = 0.7461, ROC AUC Score = 0.9092

Row 220000 (HardSampling): F1 Score = 0.7292, MCC Score = 0.7452, ROC AUC Score = 0.9069

Row 230000 (HardSampling): F1 Score = 0.7338, MCC Score = 0.7495, ROC AUC Score = 0.9074

Row 240000 (HardSampling): F1 Score = 0.7383, MCC Score = 0.7538, ROC AUC Score = 0.9089

Row 250000 (HardSampling): F1 Score = 0.7428, MCC Score = 0.7581, ROC AUC Score = 0.9098

Row 260000 (HardSampling): F1 Score = 0.7433, MCC Score = 0.7588, ROC AUC Score = 0.9091

Row 270000 (HardSampling): F1 Score = 0.7450, MCC Score = 0.7604, ROC AUC Score = 0.9099

Row 280000 (HardSampling): F1 Score = 0.7392, MCC Score = 0.7551, ROC AUC Score = 0.9071
--- ARFClassifier with HardSamplingClassifier ---

Final Scores with HardSamplingClassifier:
Final F1 Score = 0.7371
Final MCC Score = 0.7534
Final ROC AUC Score = 0.9068
```

AdaBoost Classifier Model with HardSampling

```python
# @title AdaBoost Classifier Model with HardSampling

# Instantiate the base ARFClassifier
base_model = AdaBoostClassifier(model=StandardScaler() |  HoeffdingTreeClassifier(), n_models=10,seed=42)

# Instantiate HardSamplingClassifier, wrapping the base model

model_hs_ada = HardSamplingClassifier(classifier=base_model, size=100, p=0.2, seed=42)


# Instantiate the hardOS metrics
f1_hs_ada = F1()
mcc_hs_ada = MCC()
roc_hs_ada = ROCAUC()


# Separate features (X) and target (y)
X = df.drop('Class', axis=1)
y = df['Class']

# Iterate through the DataFrame
for i, (x, y_true) in enumerate(iter_pandas(X, y)):

    # Predict and update metrics for HardOS
    y_pred_proba = model_hs_ada.predict_proba_one(x)
    y_pred = model_hs_ada.predict_one(x)

    f1_hs_ada.update(y_true, y_pred)
    mcc_hs_ada.update(y_true, y_pred)


    if y_pred_proba: # Check if predict_proba_one returned probabilities
         roc_hs_ada.update(y_true, y_pred_proba)


    # Learn from the true label using the HardSamplingClassifier
    model_hs_ada.learn_one(x, y_true)

    # Print scores every 10000 rows
    if (i + 1) % 10000 == 0:
        print(f"\nRow {i+1} (HardSampling): F1 Score = {f1_hs_ada.get():.4f}, MCC Score = {mcc_hs_ada.get():.4f}, ROC AUC Score = {roc_hs_ada.get():.4f}")


# # --- AdaBoostClassifier with HardSamplingClassifier ---
print("--- AdaBoostClassifier with HardSamplingClassifier ---")

# Print final scores
print("\nFinal Scores with HardSamplingClassifier:")
print(f"Final F1 Score = {f1_hs_ada.get():.4f}")
print(f"Final MCC Score = {mcc_hs_ada.get():.4f}")
print(f"Final ROC AUC Score = {roc_hs_ada.get():.4f}")
```

```
Row 20000 (HardSampling): F1 Score = 0.7949, MCC Score = 0.7973, ROC AUC Score = 0.9581

Row 30000 (HardSampling): F1 Score = 0.7647, MCC Score = 0.7684, ROC AUC Score = 0.9493

Row 40000 (HardSampling): F1 Score = 0.7831, MCC Score = 0.7866, ROC AUC Score = 0.9490

Row 50000 (HardSampling): F1 Score = 0.8303, MCC Score = 0.8318, ROC AUC Score = 0.9612

Row 60000 (HardSampling): F1 Score = 0.8027, MCC Score = 0.8055, ROC AUC Score = 0.9489

Row 70000 (HardSampling): F1 Score = 0.7886, MCC Score = 0.7925, ROC AUC Score = 0.9416



Row 90000 (HardSampling): F1 Score = 0.7979, MCC Score = 0.8022, ROC AUC Score = 0.9442
```

```
Row 100000 (HardSampling): F1 Score = 0.7891, MCC Score = 0.7932, ROC AUC Score = 0.9345

Row 110000 (HardSampling): F1 Score = 0.7907, MCC Score = 0.7953, ROC AUC Score = 0.9312

Row 120000 (HardSampling): F1 Score = 0.7883, MCC Score = 0.7930, ROC AUC Score = 0.9287

Row 130000 (HardSampling): F1 Score = 0.7915, MCC Score = 0.7960, ROC AUC Score = 0.9318

Row 140000 (HardSampling): F1 Score = 0.7890, MCC Score = 0.7939, ROC AUC Score = 0.9325

Row 150000 (HardSampling): F1 Score = 0.7871, MCC Score = 0.7919, ROC AUC Score = 0.9326

Row 160000 (HardSampling): F1 Score = 0.7981, MCC Score = 0.8022, ROC AUC Score = 0.9389

Row 170000 (HardSampling): F1 Score = 0.7982, MCC Score = 0.8019, ROC AUC Score = 0.9399

Row 180000 (HardSampling): F1 Score = 0.7970, MCC Score = 0.8009, ROC AUC Score = 0.9406

Row 190000 (HardSampling): F1 Score = 0.7910, MCC Score = 0.7953, ROC AUC Score = 0.9405

Row 200000 (HardSampling): F1 Score = 0.7779, MCC Score = 0.7830, ROC AUC Score = 0.9381

Row 210000 (HardSampling): F1 Score = 0.7767, MCC Score = 0.7821, ROC AUC Score = 0.9383

Row 220000 (HardSampling): F1 Score = 0.7756, MCC Score = 0.7814, ROC AUC Score = 0.9381

Row 230000 (HardSampling): F1 Score = 0.7769, MCC Score = 0.7828, ROC AUC Score = 0.9385

Row 240000 (HardSampling): F1 Score = 0.7768, MCC Score = 0.7829, ROC AUC Score = 0.9398

Row 250000 (HardSampling): F1 Score = 0.7768, MCC Score = 0.7831, ROC AUC Score = 0.9410

Row 260000 (HardSampling): F1 Score = 0.7760, MCC Score = 0.7822, ROC AUC Score = 0.9389

Row 270000 (HardSampling): F1 Score = 0.7775, MCC Score = 0.7835, ROC AUC Score = 0.9390

Row 280000 (HardSampling): F1 Score = 0.7705, MCC Score = 0.7763, ROC AUC Score = 0.9364
--- AdaBoostClassifier with HardSamplingClassifier ---

Final Scores with HardSamplingClassifier:
Final F1 Score = 0.7698
Final MCC Score = 0.7757
```

```
1 Start coding or generate with AI.
```

Drift Enabled Modelling

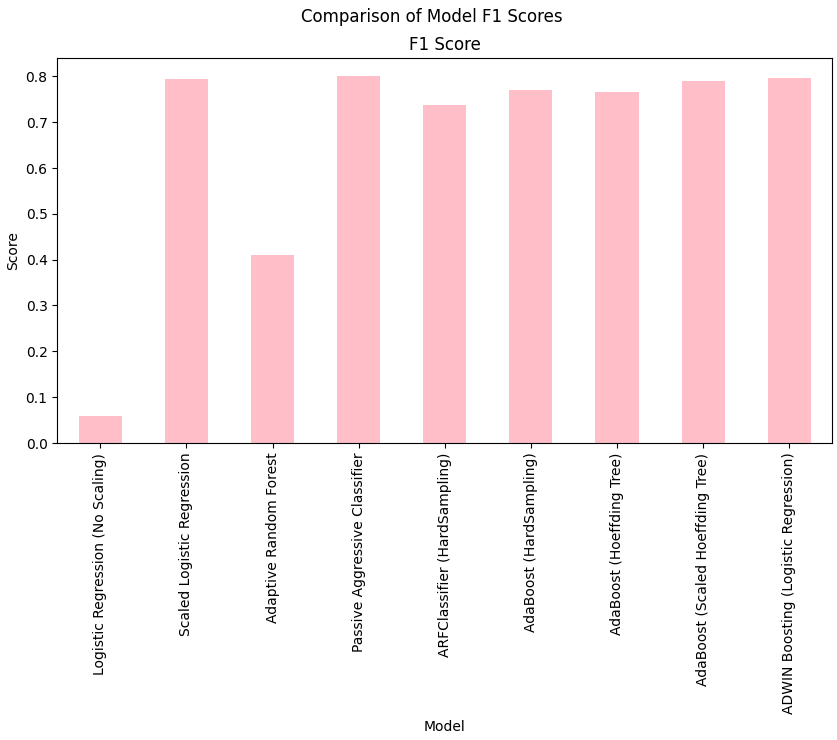
```
# Instantiate the ADWINBoostingClassifier
lr_adwin_booster = ADWINBoostingClassifier(model=StandardScaler() | LogisticRegression(), n_models=10,seed=42)

# Instantiate the metrics
f1_adwin_lr = F1()
mcc_adwin_lr = MCC()
roc_adwin_lr = ROCAUC()

# Separate features (X) and target (y)
X_adwin = df.drop('Class', axis=1)
y_true_adwin = df['Class']

# Iterate through the DataFrame instance by instance
for i, (x, y_true) in enumerate(iter_pandas(X_adwin, y_true_adwin)):

    # Predict and update metrics
    y_pred_proba = lr_adwin_booster.predict_proba_one(x)
    y_pred = lr_adwin_booster.predict_one(x)

    f1_adwin_lr.update(y_true, y_pred)
    mcc_adwin_lr.update(y_true, y_pred)

    # Only update ROC AUC if the model provides probability estimates and they are not None or empty
    if y_pred_proba:
         roc_adwin_lr.update(y_true, y_pred_proba)

    # Learn from the true label
    lr_adwin_booster.learn_one(x, y_true)

    # Print scores every 10000 rows
    if (i + 1) % 10000 == 0:
        print(f"Row {i+1} (ADWIN Boosting): F1 Score = {f1_adwin_lr.get():.4f}, MCC Score = {mcc_adwin_lr.get():.4f}, ROC AUC Score = {roc_adwin_lr.get():.4f}")

# Print final scores
print("Final Scores with ADWIN Boosting Classifier:")
print(f"Final F1 Score = {f1_adwin_lr.get():.4f}")
print(f"Final MCC Score = {mcc_adwin_lr.get():.4f}")
print(f"Final ROC AUC Score = {roc_adwin_lr.get():.4f}")
```

```
Row 10000 (ADWIN Boosting): F1 Score = 0.8533, MCC Score = 0.8529, ROC AUC Score = 0.9460
Row 20000 (ADWIN Boosting): F1 Score = 0.7485, MCC Score = 0.7481, ROC AUC Score = 0.9160
Row 30000 (ADWIN Boosting): F1 Score = 0.7241, MCC Score = 0.7257, ROC AUC Score = 0.8923
Row 40000 (ADWIN Boosting): F1 Score = 0.7396, MCC Score = 0.7416, ROC AUC Score = 0.8981
Row 50000 (ADWIN Boosting): F1 Score = 0.7899, MCC Score = 0.7914, ROC AUC Score = 0.9216
Row 60000 (ADWIN Boosting): F1 Score = 0.7703, MCC Score = 0.7737, ROC AUC Score = 0.9073
Row 70000 (ADWIN Boosting): F1 Score = 0.7697, MCC Score = 0.7734, ROC AUC Score = 0.9051
Row 80000 (ADWIN Boosting): F1 Score = 0.7831, MCC Score = 0.7869, ROC AUC Score = 0.9077
Row 90000 (ADWIN Boosting): F1 Score = 0.7937, MCC Score = 0.7975, ROC AUC Score = 0.9119
Row 100000 (ADWIN Boosting): F1 Score = 0.7910, MCC Score = 0.7954, ROC AUC Score = 0.9054
Row 110000 (ADWIN Boosting): F1 Score = 0.7925, MCC Score = 0.7974, ROC AUC Score = 0.9013
Row 120000 (ADWIN Boosting): F1 Score = 0.7865, MCC Score = 0.7910, ROC AUC Score = 0.8984
Row 130000 (ADWIN Boosting): F1 Score = 0.7768, MCC Score = 0.7821, ROC AUC Score = 0.8981
```

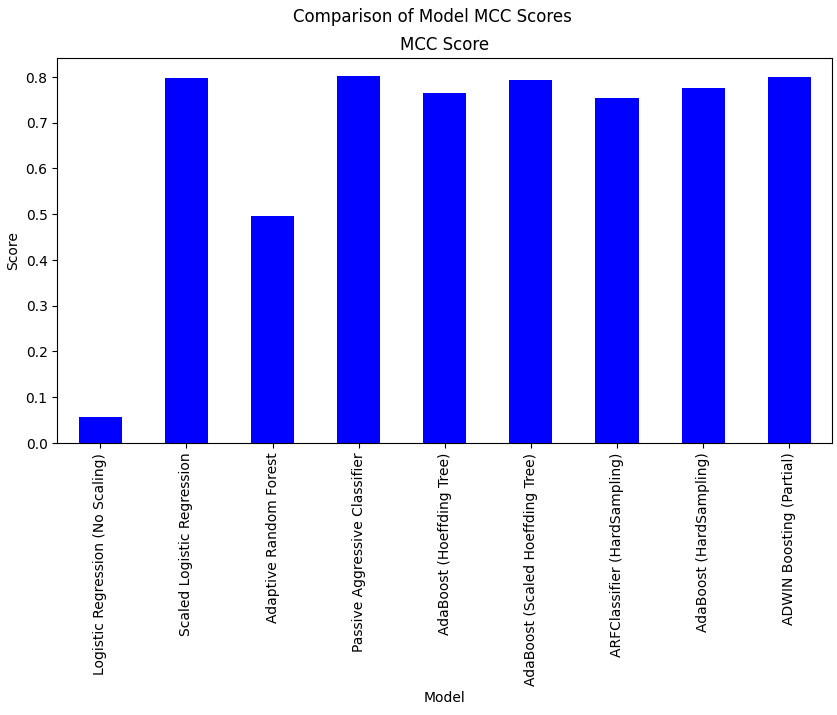
```
Row 140000 (ADWIN Boosting): F1 Score = 0.7771, MCC Score = 0.7825, ROC AUC Score = 0.8974
Row 150000 (ADWIN Boosting): F1 Score = 0.7757, MCC Score = 0.7804, ROC AUC Score = 0.8973
Row 160000 (ADWIN Boosting): F1 Score = 0.8031, MCC Score = 0.8071, ROC AUC Score = 0.9084
Row 170000 (ADWIN Boosting): F1 Score = 0.8006, MCC Score = 0.8046, ROC AUC Score = 0.9094
Row 180000 (ADWIN Boosting): F1 Score = 0.7994, MCC Score = 0.8036, ROC AUC Score = 0.9091
Row 190000 (ADWIN Boosting): F1 Score = 0.7946, MCC Score = 0.7986, ROC AUC Score = 0.9095
Row 200000 (ADWIN Boosting): F1 Score = 0.7914, MCC Score = 0.7957, ROC AUC Score = 0.9062
Row 210000 (ADWIN Boosting): F1 Score = 0.7927, MCC Score = 0.7967, ROC AUC Score = 0.9071
Row 220000 (ADWIN Boosting): F1 Score = 0.7913, MCC Score = 0.7956, ROC AUC Score = 0.9049
Row 230000 (ADWIN Boosting): F1 Score = 0.7926, MCC Score = 0.7969, ROC AUC Score = 0.9055
Row 240000 (ADWIN Boosting): F1 Score = 0.7949, MCC Score = 0.7991, ROC AUC Score = 0.9060
Row 250000 (ADWIN Boosting): F1 Score = 0.7971, MCC Score = 0.8013, ROC AUC Score = 0.9070
Row 260000 (ADWIN Boosting): F1 Score = 0.7976, MCC Score = 0.8020, ROC AUC Score = 0.9053
Row 270000 (ADWIN Boosting): F1 Score = 0.7991, MCC Score = 0.8032, ROC AUC Score = 0.9062
Row 280000 (ADWIN Boosting): F1 Score = 0.7950, MCC Score = 0.7992, ROC AUC Score = 0.9035
Final Scores with ADWIN Boosting Classifier:
Final F1 Score = 0.7955
Final MCC Score = 0.7997
Final ROC AUC Score = 0.9032
```

```
Start coding or generate with AI.
```

```
# Collected final scores for each model from the metric variables
results = {
    'Model': [
        'Logistic Regression (No Scaling)',
        'Scaled Logistic Regression',
        'Adaptive Random Forest',
        'Passive Aggressive Classifier',
        'ARFClassifier (HardSampling)',
        'AdaBoost (HardSampling)',
        'AdaBoost (Hoeffding Tree)',
        'AdaBoost (Scaled Hoeffding Tree)',
        'ADWIN Boosting (Logistic Regression)'
    ],
    'F1 Score': [
        ns_logreg_f1.get(),
        logreg_f1.get(),
        ARF_f1.get(),
        f1_pa.get(),
        f1_hs.get(),
        f1_hs_ada.get(),
        f1_ada.get(),
        f1_ada_sc.get(),
        f1_adwin_lr.get()
    ],
    'MCC Score': [
        ns_logreg_mcc.get(),
        logreg_mcc.get(),
        ARF_mcc.get(),
        mcc_pa.get(),
        mcc_hs.get(),
        mcc_hs_ada.get(),
        mcc_ada.get(),
        mcc_ada_sc.get(),
        mcc_adwin_lr.get()
    ],
    'ROC AUC Score': [
        ns_logreg_roc.get(),
        logreg_roc.get(),
        ARF_roc.get(),
        roc_pa.get(),
        roc_hs.get(),
        roc_hs_ada.get(),
        roc_ada.get(),
        roc_ada_sc.get(),
        roc_adwin_lr.get()
    ]
}

df_results = pd.DataFrame(results)

# Set the index to model names for easier plotting
df_results = df_results.set_index('Model')

# Plotting the scores
fig, ax = plt.subplots(figsize=(10, 5))
fig.suptitle('Comparison of Model F1 Scores')

# F1 Score plot
df_results['F1 Score'].plot(kind='bar', ax=ax, color='pink')
ax.set_title('F1 Score')
ax.set_ylabel('Score')
ax.set_xlabel('Model')
ax.tick_params(axis='x')

plt.show()
```

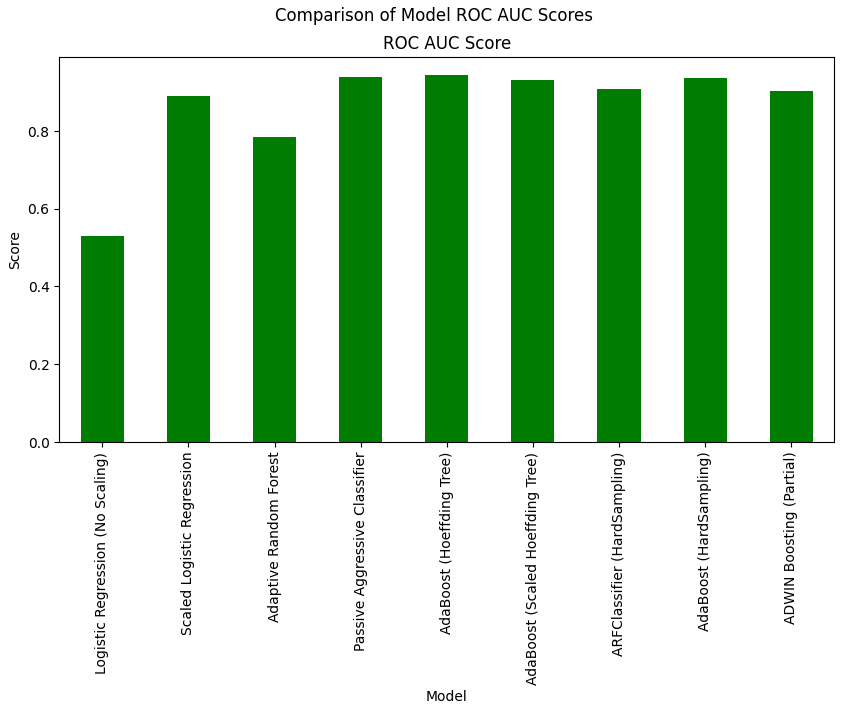



```
# Plotting only the MCC scores
fig, ax = plt.subplots(figsize=(10, 5))
fig.suptitle('Comparison of Model MCC Scores')

# MCC Score plot
df_results['MCC Score'].plot(kind='bar', ax=ax, color='blue')
ax.set_title('MCC Score')
ax.set_ylabel('Score')
ax.set_xlabel('Model')
ax.tick_params(axis='x')

plt.show()
```




```
# Plotting only the ROC AUC scores
fig, ax = plt.subplots(figsize=(10, 5))
fig.suptitle('Comparison of Model ROC AUC Scores')

# ROC AUC Score plot
df_results['ROC AUC Score'].plot(kind='bar', ax=ax, color='green')
ax.set_title('ROC AUC Score')
ax.set_ylabel('Score')
ax.set_xlabel('Model')
```

```
ax.tick_params(axis='x')

plt.show()
```




```
Start coding or generate with AI.
```