

Supervised Learning Final Project1

October 10, 2025

Supervised Learning Final Project - A Comparative Study of Random Forest and SVM Models

Author: Wenhao Chen

UCI ML data repository: devanagari handwritten character dataset

Data Understand

This is an image database of Handwritten Devanagari characters. It contains handwritten samples of Devanagari script, which is widely used in Indian languages such as Hindi, Sanskrit, Marathi, and Nepali. Recognizing specific characters can enhance the accuracy of computer vision programs, and the resulting technology can be applied to translation, text-to-speech, and accessibility tools.

There are 46 classes of characters with 2000 examples each. The dataset is split into training set(85%) and testing set(15%) No missing Values

- Total Classes: 46
- 36 characters (-, etc.)
- 10 numerals (-)
- Total Samples: ~92,000 images
- Image Size: 32 × 32 pixels (grayscale)
- Format: Each sample is labeled with the corresponding character/numeral.
- Store:

```
thisfile.ipynb  
dataset/  
    train/  
        digit_0/  
            img1.png  
            img2.png  
        digit_1/  
        character_1_ka/  
        character_2_kha/  
        ...  
    test/  
        digit_0/  
        digit_1/  
        character_1_ka/  
        character_2_kha/  
        ...
```

Identifying a Supervised Machine Learning Problem

this is Multiclass Classification Supervised Learning Problem input is handwritten grayscale images of size 32×32 . output is one of 46 possible labels

```
[1]: import os
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.datasets import load_files
from sklearn.metrics import mean_squared_error
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
```

```
[2]: path = "./DevanagariHandwrittenCharacterDataset/"
testpath = f"[path]test"
trainpath = f"[path]train"
```

EDA:

```
[3]: def load_images_from_folder(folder, img_size=32):
    data = load_files(folder)
    file_paths = np.array(data['filenames'])
    target = np.array(data['target'])
    target_names = data['target_names']

    images = []
    for f in file_paths:
        img = cv2.imread(f, cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (img_size, img_size))
        images.append(img.flatten())

    return np.array(images), target, target_names

def view_image(image, label=None, classname=None):
    """
    function to plot digit examples
```

```

"""
image = image.reshape(32, 32)
label = label
plt.imshow(image, cmap="gray")
plt.title(f"Label: {label} ({classname})")
plt.axis("off")
plt.show()

```

[4]: train, target, targetnames = load_images_from_folder(trainpath)
test, testtarget, _ = load_images_from_folder(testpath)

[8]: print(target, targetnames)
labeltrans = {i:name.split('_')[-1] for i, name in enumerate(targetnames)}
print(labeltrans)

```

[15 26 0 ... 25 25 40] ['character_10_yna', 'character_11_taamatar',
'character_12_thaa', 'character_13_daa', 'character_14_dhaa',
'character_15_adna', 'character_16_tabala', 'character_17 tha',
'character_18_da', 'character_19_dha', 'character_1_ka', 'character_20_na',
'character_21_pa', 'character_22_pha', 'character_23_ba', 'character_24_bha',
'character_25_ma', 'character_26_yaw', 'character_27_ra', 'character_28_la',
'character_29_waw', 'character_2_kha', 'character_30_motosaw',
'character_31_petchiryakha', 'character_32_patalosaw', 'character_33_ha',
'character_34_chhya', 'character_35_tra', 'character_36_gya', 'character_3_ga',
'character_4_gha', 'character_5_kna', 'character_6_cha', 'character_7_chha',
'character_8_ja', 'character_9_jha', 'digit_0', 'digit_1', 'digit_2', 'digit_3',
'digit_4', 'digit_5', 'digit_6', 'digit_7', 'digit_8', 'digit_9']
{0: 'yna', 1: 'taamatar', 2: 'thaa', 3: 'daa', 4: 'dhaa', 5: 'adna', 6:
'tabala', 7: 'tha', 8: 'da', 9: 'dha', 10: 'ka', 11: 'na', 12: 'pa', 13: 'pha',
14: 'ba', 15: 'bha', 16: 'ma', 17: 'yaw', 18: 'ra', 19: 'la', 20: 'waw', 21:
'kha', 22: 'motosaw', 23: 'petchiryakha', 24: 'patalosaw', 25: 'ha', 26:
'chhya', 27: 'tra', 28: 'gya', 29: 'ga', 30: 'gha', 31: 'kna', 32: 'cha', 33:
'chha', 34: 'ja', 35: 'jha', 36: '0', 37: '1', 38: '2', 39: '3', 40: '4', 41:
'5', 42: '6', 43: '7', 44: '8', 45: '9'}

```

[10]: view_image(train[31], target[31], labeltrans[target[31]])

Label: 45 (9)



```
[18]: unique_classes = np.unique(target)
indices = [np.random.choice(np.where(target == cls)[0]) for cls in_
           ↪unique_classes]
plt.figure(figsize=(15, 10))

for i, idx in enumerate(indices):
    image = train[idx].reshape(32, 32)
    label = target[idx]

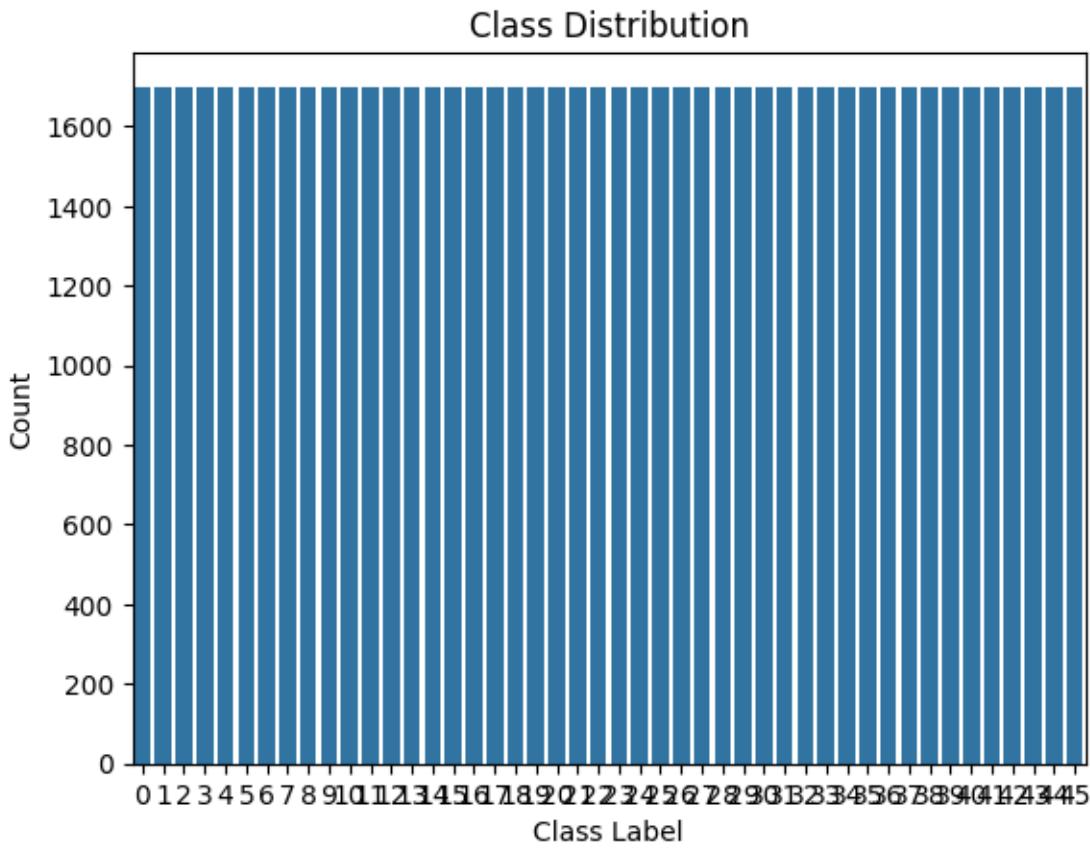
    plt.subplot(6, 8, i+1)
    plt.imshow(image, cmap="gray")
    plt.title(labeltrans[label], fontsize=12)
    plt.axis("off")

plt.tight_layout()
plt.show()
```



from the cell above, i plot one of each from 46 type

```
[17]: sns.countplot(x=target)
plt.title("Class Distribution")
plt.xlabel("Class Label")
plt.ylabel("Count")
plt.show()
```



Fit Model

the Data is Banlanced, so we don't need to apply SMOTE.

AdaBoost

```
[58]: ada = AdaBoostClassifier(estimator=DecisionTreeClassifier(max_depth=1,
    ↪random_state=114514), n_estimators=200, random_state=114514)
ada.fit(train, target)

y_pred_ada = ada.predict(test)
print("AdaBoost Accuracy:", accuracy_score(testtarget, y_pred_ada))
```

AdaBoost Accuracy: 0.2086231884057971

The accuracy of the model is 0.21, which aligns with my expectations. This result is reasonable given that AdaBoost is a relatively simple ensemble method that works by sequentially combining multiple weak learners. typically shallow decision trees—to gradually improve performance. Its simplicity and reliance on weak classifiers often lead to modest accuracy, especially when applied to complex datasets.

random forest

using the GridSearchCV search the best hyperparameter for this dataset(a lot of output in this cell. so i comment it). and we have best parameter: {'max_depth': 25, 'max_features': 'sqrt', 'n_estimators': 500}

```
[ ]: """
param_grid = {
    "n_estimators": [100, 300, 500],
    "max_depth": [15, 25, 35],
    "max_features": ["sqrt", "log2"]
}
#Best params: {'max_depth': 25, 'max_features': 'log2', 'n_estimators': 500}
rf = RandomForestClassifier(random_state=114514, n_jobs=-1)

grid = GridSearchCV(rf, param_grid, cv=3, scoring="accuracy", n_jobs=1, verbose=2)
grid.fit(train, target)

print("Best params:", grid.best_params_)
print("Best CV accuracy:", grid.best_score_)
print("Test accuracy:", grid.score(test, testtarget))
"""

```

```
[81]: rf = RandomForestClassifier(
    n_estimators=500,
    max_depth=25,
    max_features="sqrt",
    n_jobs=-1,
    random_state=114514
)
rf.fit(train, target)
```

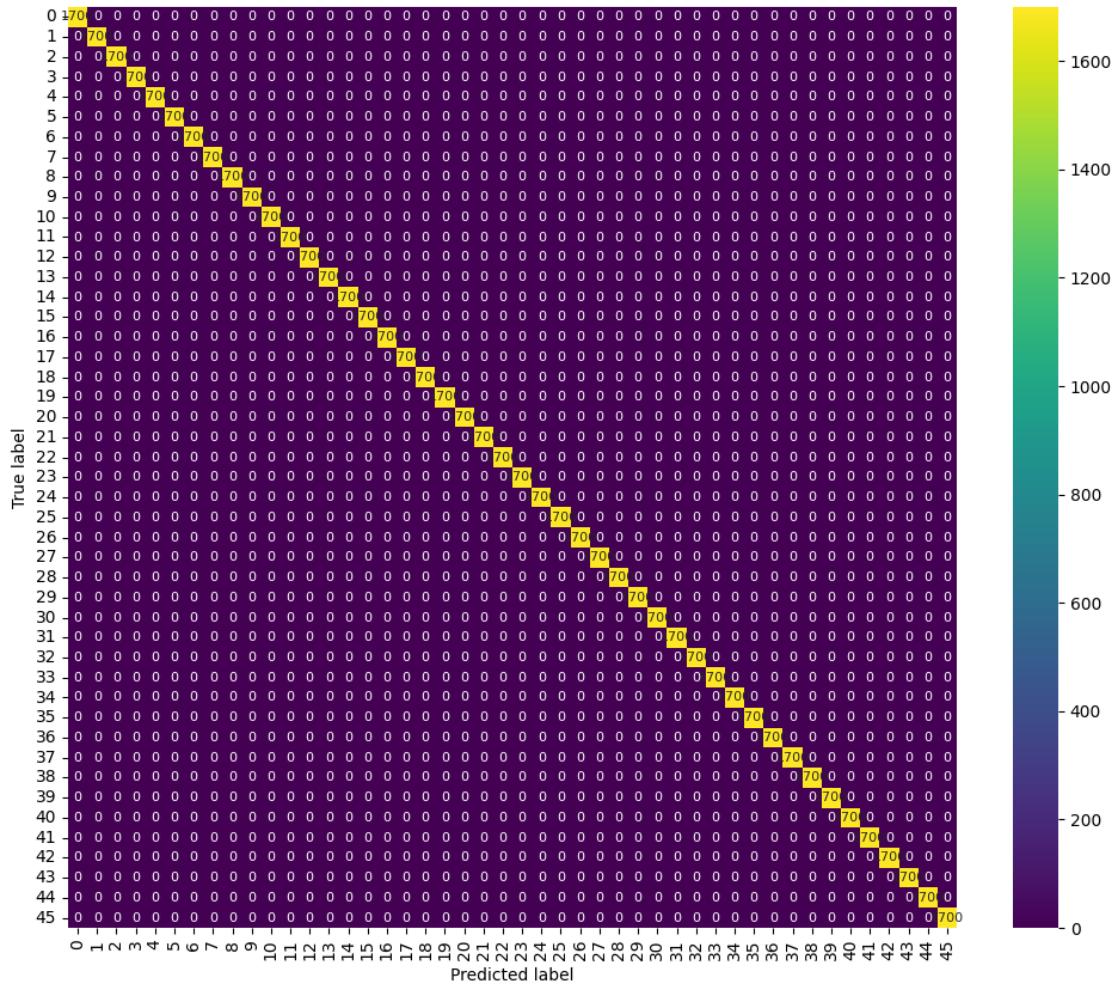
```
[81]: RandomForestClassifier(max_depth=25, n_estimators=500, n_jobs=-1,
                             random_state=114514)
```

```
[82]: y_trainpred = rf.predict(train)
print("Random Forest Accuracy:", accuracy_score(target, y_trainpred))
```

```
Random Forest Accuracy: 1.0
```

```
[83]: conf_matrix = confusion_matrix(target, y_trainpred)

plt.figure(figsize=(12, 10))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="viridis",
            annot_kws={"size":8})
plt.xlabel("Predicted label")
plt.ylabel("True label")
plt.show()
```



```
[84]: test1 = y_trainpred[y_trainpred == 0]
len(test1)
```

[84]: 1700

The Random Forest model fit the training data exceptionally well, achieving a perfect accuracy score of 1.0. This indicates that the ensemble of decision trees was able to capture all patterns and distinctions within the training set without any misclassifications. While such a result demonstrates the model's strong learning capacity, it may also suggest potential overfitting: especially if the test accuracy is significantly lower, since perfect performance on training data doesn't always translate to generalization on unseen data.

```
[67]: y_testpred = rf.predict(test)
print("Random Forest Accuracy:", accuracy_score(testtarget, y_testpred))
test_matrix = confusion_matrix(testtarget, y_testpred)

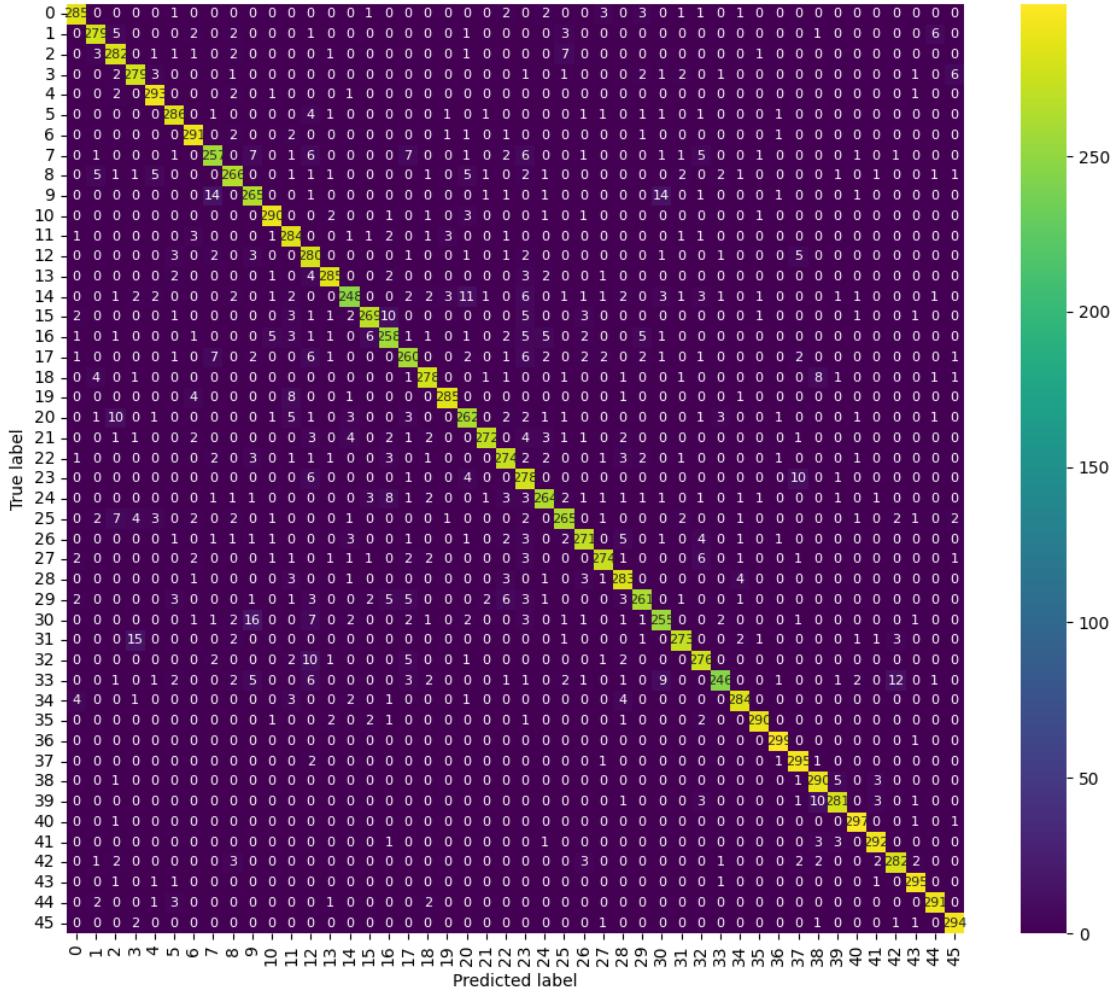
plt.figure(figsize=(12, 10))
```

```

sns.heatmap(test_matrix, annot=True, fmt="d", cmap="viridis",
            annot_kws={"size":8})
plt.xlabel("Predicted label")
plt.ylabel("True label")
plt.show()

```

Random Forest Accuracy: 0.9249275362318841



```

[71]: mse = mean_squared_error(testtarget, y_testpred)
print("MSE:", mse)

```

```
print(classification_report(testtarget, y_testpred))
```

MSE: 18.864202898550726

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.95	0.95	0.95	300
---	------	------	------	-----

1	0.94	0.93	0.93	300
2	0.89	0.94	0.91	300
3	0.91	0.93	0.92	300
4	0.94	0.98	0.96	300
5	0.93	0.95	0.94	300
6	0.94	0.97	0.95	300
7	0.89	0.86	0.87	300
8	0.92	0.89	0.90	300
9	0.87	0.88	0.88	300
10	0.95	0.97	0.96	300
11	0.89	0.95	0.92	300
12	0.81	0.93	0.87	300
13	0.95	0.95	0.95	300
14	0.92	0.83	0.87	300
15	0.94	0.90	0.92	300
16	0.88	0.86	0.87	300
17	0.88	0.87	0.87	300
18	0.94	0.93	0.93	300
19	0.97	0.95	0.96	300
20	0.88	0.87	0.88	300
21	0.97	0.91	0.94	300
22	0.90	0.91	0.91	300
23	0.82	0.93	0.87	300
24	0.92	0.88	0.90	300
25	0.92	0.88	0.90	300
26	0.93	0.90	0.92	300
27	0.95	0.91	0.93	300
28	0.91	0.94	0.92	300
29	0.93	0.87	0.90	300
30	0.88	0.85	0.87	300
31	0.95	0.91	0.93	300
32	0.90	0.92	0.91	300
33	0.95	0.82	0.88	300
34	0.95	0.95	0.95	300
35	0.98	0.97	0.97	300
36	0.97	1.00	0.99	300
37	0.92	0.98	0.95	300
38	0.92	0.97	0.94	300
39	0.95	0.94	0.94	300
40	0.97	0.99	0.98	300
41	0.96	0.97	0.97	300
42	0.94	0.94	0.94	300
43	0.96	0.98	0.97	300
44	0.96	0.97	0.97	300
45	0.96	0.98	0.97	300
accuracy			0.92	13800
macro avg	0.93	0.92	0.92	13800

weighted avg	0.93	0.92	0.92	13800
--------------	------	------	------	-------

Result on RandomForest

- Overall Model Performance
 - Accuracy: ~92.5% — the model correctly predicts about 9 out of 10 characters.
 - F1-score (macro & weighted): ~0.92–0.93 — strong balance between precision and recall across classes.
 - MSE: ~18.86 — on average, there are misclassification errors, but since MSE is less interpretable in classification, it shows the model is not perfect.

Conclusion: The Random Forest provides high and stable performance, showing it is a strong baseline model.

2. Class-wise Observations

Looking at precision, recall, and F1 per class:

- Strong classes (~0.95 F1): digits (0, 4, 6, 10, 13, 19, 34–45) → model distinguishes these well.
- Weaker classes (~0.85–0.90 F1): e.g., 7 (chha), 9 (jha), 12 (thaa), 14 (dhaa), 16 (tabala), 17 (tha), 20 (na), 23 (ba), 30 (motosaw), 33 (ha) → these characters are harder to classify, often confused with similar-looking symbols.

These weaker classes likely share visual similarities with other characters, leading to misclassifications.

- Interpretation of Errors
 - High accuracy but not perfect → RF slightly struggles with fine grained visual distinctions.
 - MSE is non-zero → indicates presence of mislabels or characters with overlapping patterns.

The Random Forest classifier achieved an overall accuracy of 92.5% with a macro-average F1-score of 0.92, showing strong performance on the Devanagari handwritten character dataset. Most classes, especially the digits and several core characters, were classified with high precision and recall (>95%). However, some visually similar characters such as jha, thaa, and dhaa were more challenging, with F1-scores around 0.85–0.90, suggesting frequent misclassifications. The mean squared error (18.86) further indicates that while the model performs well,

SVM

```
[29]: param_grid = {
    'loss': ['hinge', 'log_loss'],
    'alpha': [1e-4, 1e-3, 1e-2],
    'penalty': ['l2', 'l1', 'elasticnet'],
    'learning_rate': ['optimal', 'adaptive'],
    'eta0': [0.001, 0.01, 0.1]
}
scaler = StandardScaler()
sgd = SGDClassifier(random_state=114514, max_iter=1000, tol=1e-3)
grid = GridSearchCV(
    sgd,
    param_grid,
    cv=3,
    n_jobs=-1,
    verbose=2,
    scoring='accuracy'
```

```

)
train_scaled = scaler.fit_transform(train)
test_scaled = scaler.transform(test)
grid.fit(train_scaled[:10000], target[:10000])

print("Best params:", grid.best_params_)

#Best params: {'alpha': 0.0001, 'learning_rate': 'optimal', 'loss': 'hinge', ↴
    ↴'penalty': 'l2'}

```

Fitting 3 folds for each of 108 candidates, totalling 324 fits

Best params: {'alpha': 0.01, 'eta0': 0.01, 'learning_rate': 'adaptive', 'loss': 'log_loss', 'penalty': 'l2'}

[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=log_loss, penalty=l2; total time= 1.1min

[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=log_loss, penalty=elasticnet; total time= 1.8min

[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=log_loss, penalty=l2; total time= 41.5s

[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=log_loss, penalty=elasticnet; total time= 1.0min

[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=elasticnet; total time= 1.6min

[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=hinge, penalty=l2; total time= 25.3s

[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=hinge, penalty=l1; total time= 52.4s

[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=log_loss, penalty=l2; total time= 53.6s

[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=log_loss, penalty=elasticnet; total time= 1.3min

[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=log_loss, penalty=l2; total time= 1.1min

[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=log_loss, penalty=elasticnet; total time= 2.1min

[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=log_loss, penalty=l1; total time= 2.6min

[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=hinge, penalty=l2; total time= 20.1s

[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=hinge, penalty=elasticnet; total time= 37.2s

[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=log_loss, penalty=l1; total time= 1.4min

[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=elasticnet; total time= 48.7s

[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=log_loss, penalty=elasticnet; total time= 1.3min

[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=log_loss,

```

penalty=l1; total time= 1.4min
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=l2; total time= 51.2s
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=l2;
total time= 32.3s
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 1.0min
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 2.3min
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.3min
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=hinge, penalty=l1;
total time= 27.7s
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 1.1min
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=l2;
total time= 34.0s
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 40.1s
[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=hinge,
penalty=l2; total time= 47.5s
[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 3.9min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=l2; total time= 54.1s
[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 1.7min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 54.5s
[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 1.6min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 4.0min
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=hinge, penalty=l1;
total time= 1.6min
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 1.3min
[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.2min
[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=l2; total time= 50.4s
[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=hinge, penalty=l2;
total time= 22.2s
[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 43.3s
[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 1.4min
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=hinge,

```

```

penalty=elasticnet; total time= 48.6s
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 1.4min
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 1.9min
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=hinge, penalty=l2;
total time= 18.4s
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=hinge, penalty=l1;
total time= 19.4s
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 1.2min
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=hinge, penalty=l2;
total time= 19.1s
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 34.1s
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=l2;
total time= 9.5s
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=l1;
total time= 21.2s
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=log_loss, penalty=l2;
total time= 21.0s
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=l2;
total time= 33.4s
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 41.3s
[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=hinge,
penalty=l2; total time= 53.3s
[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 4.1min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 4.2min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 1.6min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=l2;
total time= 34.7s
[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=l1;
total time= 1.2min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=l2; total time= 1.2min
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=hinge, penalty=l2;
total time= 23.7s
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 47.1s
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=l2; total time= 51.0s
[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=l1; total time= 27.3s
[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=log_loss,

```

```

penalty=12; total time= 41.0s
[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=12;
total time= 23.1s
[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=11;
total time= 1.6min
[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=hinge, penalty=12;
total time= 23.4s
[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 43.7s
[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.5min
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=11; total time= 3.1min
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 20.4s
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 46.8s
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 26.7s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=12;
total time= 9.7s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=11;
total time= 19.1s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=12; total time= 20.5s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 49.8s
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=11; total time= 1.1min
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=12;
total time= 32.2s
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 50.4s
[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 1.7min
[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=12; total time= 20.3s
[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=11; total time= 39.9s
[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 39.9s
[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=11; total time= 1.1min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=11;
total time= 4.0min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=12;
total time= 49.8s
[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=hinge,

```

```

penalty=elasticnet; total time= 1.7min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=l2;
total time= 32.3s
[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=l1;
total time= 55.4s
[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 56.6s
[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.9min
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 3.1min
[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 3.1min
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=l2; total time= 52.3s
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=l1;
total time= 51.3s
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 2.0min
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=hinge, penalty=l2;
total time= 17.6s
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=hinge, penalty=l1;
total time= 21.1s
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 21.3s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=l2;
total time= 9.6s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=l1;
total time= 18.8s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=l2; total time= 20.0s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 48.3s
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=l2; total time= 39.7s
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 23.2s
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 47.6s
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=l2; total time= 1.1min
[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 1.6min
[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=l2; total time= 20.3s
[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=l1; total time= 41.1s
[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=hinge,

```

```

penalty=elasticnet; total time= 38.9s
[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 1.2min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=l1;
total time= 3.9min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.3min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=l2; total time= 1.2min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=l2;
total time= 34.5s
[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=l1;
total time= 1.2min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=l2; total time= 1.2min
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=hinge, penalty=l1;
total time= 1.6min
[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=l2; total time= 20.0s
[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=l1; total time= 27.5s
[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=l2; total time= 40.7s
[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=l2;
total time= 22.0s
[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=l1;
total time= 1.6min
[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=hinge, penalty=l1;
total time= 34.1s
[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=l2; total time= 44.4s
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=l2;
total time= 25.1s
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=l1;
total time= 1.6min
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=l1;
total time= 51.1s
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=l2; total time= 1.1min
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=hinge, penalty=l2;
total time= 10.3s
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=hinge, penalty=l1;
total time= 19.0s
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=l2; total time= 20.7s
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 48.9s
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=log_loss,

```

```

penalty=l2; total time= 39.4s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 23.0s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 1.1min
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=l2;
total time= 9.5s
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=l1;
total time= 18.4s
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 20.1s
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=log_loss, penalty=l1;
total time= 52.4s
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=l2; total time= 1.1min
[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=l2; total time= 1.1min
[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 2.0min
[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=l2; total time= 40.7s
[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.0min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 3.6min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=l2;
total time= 43.7s
[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 1.5min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 2.1min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 2.6min
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 1.4min
[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.2min
[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=l2; total time= 50.6s
[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=hinge, penalty=l2;
total time= 23.0s
[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 44.0s
[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 1.4min
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 48.9s
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=l2;

```

```

total time= 32.7s
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 1.0min
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 2.3min
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 1.3min
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=hinge, penalty=l2;
total time= 19.0s
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 34.2s
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=l2;
total time= 8.9s
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=l1;
total time= 19.1s
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=log_loss, penalty=l2;
total time= 20.6s
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 49.0s
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=l2; total time= 1.1min
[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=hinge,
penalty=l1; total time= 3.9min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=l2;
total time= 50.1s
[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=l2; total time= 1.2min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=hinge,
penalty=l2; total time= 25.3s
[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=hinge,
penalty=l1; total time= 52.3s
[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 53.9s
[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 1.5min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 3.6min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.9min
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 3.1min
[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 3.1min
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=l2; total time= 51.3s
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=l1;
total time= 50.4s
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=log_loss,

```

```

penalty=12; total time= 1.1min
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=hinge, penalty=12;
total time= 8.4s
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=hinge, penalty=11;
total time= 17.3s
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=12; total time= 17.8s
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=11; total time= 55.3s
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=12; total time= 39.9s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 20.2s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=11; total time= 53.0s
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=12; total time= 40.4s
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 22.6s
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=log_loss, penalty=11;
total time= 56.1s
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=11; total time= 1.3min
[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=hinge,
penalty=11; total time= 3.8min
[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.0min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=11; total time= 3.5min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.3min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=12; total time= 54.6s
[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 1.7min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 1.2min
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=hinge, penalty=12;
total time= 22.6s
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=hinge, penalty=11;
total time= 1.6min
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 1.4min
[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.2min
[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=12; total time= 51.9s
[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=hinge, penalty=11;

```

```

total time= 34.4s
[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=l2; total time= 43.6s
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=l2;
total time= 22.6s
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=l1;
total time= 1.6min
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=l2;
total time= 33.0s
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 1.0min
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 2.3min
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.3min
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=hinge, penalty=l1;
total time= 27.8s
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.3min
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=l1;
total time= 36.2s
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 1.3min
[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 1.5min
[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=l2; total time= 20.5s
[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=l1; total time= 39.8s
[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 39.6s
[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=l2; total time= 40.8s
[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=l2;
total time= 44.6s
[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=l2; total time= 1.1min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 2.1min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=l2; total time= 51.4s
[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=l1;
total time= 3.8min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 1.3min
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=hinge, penalty=l2;
total time= 22.6s
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=hinge,

```

```

penalty=elasticnet; total time= 46.9s
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=l2; total time= 52.6s
[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 36.8s
[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 1.4min
[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 47.2s
[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 1.4min
[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.5min
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 3.0min
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 22.9s
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 51.9s
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 26.3s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=l2;
total time= 9.8s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=l1;
total time= 21.0s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=l2; total time= 20.2s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 50.1s
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.3min
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=l1;
total time= 36.3s
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 1.3min
[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=hinge,
penalty=l1; total time= 3.9min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=l2;
total time= 41.6s
[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 1.5min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=hinge,
penalty=l2; total time= 25.1s
[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=hinge,
penalty=l1; total time= 41.4s
[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 43.3s
[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=log_loss,

```

```

penalty=l2; total time= 45.4s
[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=l1;
total time= 4.0min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=l2; total time= 1.2min
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 46.2s
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=l2; total time= 50.2s
[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=l2; total time= 20.2s
[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 37.8s
[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 1.4min
[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 47.2s
[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 1.4min
[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.5min
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 3.1min
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 23.5s
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 1.1min
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.3min
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=hinge, penalty=l1;
total time= 27.8s
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 1.1min
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 49.8s
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.6min
[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=hinge,
penalty=l2; total time= 44.8s
[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 3.8min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 1.4min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 2.1min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 1.6min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=log_loss,

```

```

penalty=l1; total time= 3.5min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.8min
[CV] END alpha=0.001, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 3.1min
[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 3.1min
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 39.7s
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 1.4min
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 2.0min
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=hinge, penalty=l2;
total time= 15.7s
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=hinge, penalty=l1;
total time= 22.7s
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=l2; total time= 39.7s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=hinge,
penalty=elasticnet; total time= 22.6s
[CV] END alpha=0.01, eta0=0.01, learning_rate=optimal, loss=log_loss,
penalty=l1; total time= 56.4s
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=l2; total time= 39.7s
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=log_loss, penalty=l2;
total time= 20.0s
[CV] END alpha=0.01, eta0=0.1, learning_rate=optimal, loss=log_loss, penalty=l1;
total time= 1.1min
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.6min
[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=l2; total time= 1.2min
[CV] END alpha=0.0001, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 2.0min
[CV] END alpha=0.0001, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 1.1min
[CV] END alpha=0.0001, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=l1;
total time= 4.0min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=l2;
total time= 41.3s
[CV] END alpha=0.0001, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=l1;
total time= 4.1min
[CV] END alpha=0.0001, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=l1; total time= 2.5min
[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=hinge,
penalty=l1; total time= 28.2s
[CV] END alpha=0.001, eta0=0.001, learning_rate=adaptive, loss=log_loss,

```

```

penalty=12; total time= 41.8s
[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=12;
total time= 22.1s
[CV] END alpha=0.001, eta0=0.01, learning_rate=optimal, loss=hinge, penalty=11;
total time= 1.6min
[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=hinge, penalty=11;
total time= 34.1s
[CV] END alpha=0.001, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=12; total time= 42.5s
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=12;
total time= 21.8s
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=hinge, penalty=11;
total time= 1.3min
[CV] END alpha=0.001, eta0=0.1, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 1.3min
[CV] END alpha=0.001, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=12; total time= 1.1min
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=hinge, penalty=12;
total time= 9.1s
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=hinge, penalty=11;
total time= 21.4s
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=12; total time= 20.9s
[CV] END alpha=0.01, eta0=0.001, learning_rate=optimal, loss=log_loss,
penalty=elasticnet; total time= 49.1s
[CV] END alpha=0.01, eta0=0.001, learning_rate=adaptive, loss=log_loss,
penalty=11; total time= 1.2min
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=hinge, penalty=12;
total time= 19.2s
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=hinge,
penalty=elasticnet; total time= 27.2s
[CV] END alpha=0.01, eta0=0.01, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.3min
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=hinge, penalty=11;
total time= 36.8s
[CV] END alpha=0.01, eta0=0.1, learning_rate=adaptive, loss=log_loss,
penalty=elasticnet; total time= 1.6min

```

[30]:

```

svcmodel = grid.best_estimator_
svcmodel.fit(train, target)
svctrainpred = svcmodel.predict(train)
print("svc Accuracy:", accuracy_score(target, svctrainpred))

```

svc Accuracy: 0.6970460358056266

[31]:

```

svctrain_matrix = confusion_matrix(target, svctrainpred)

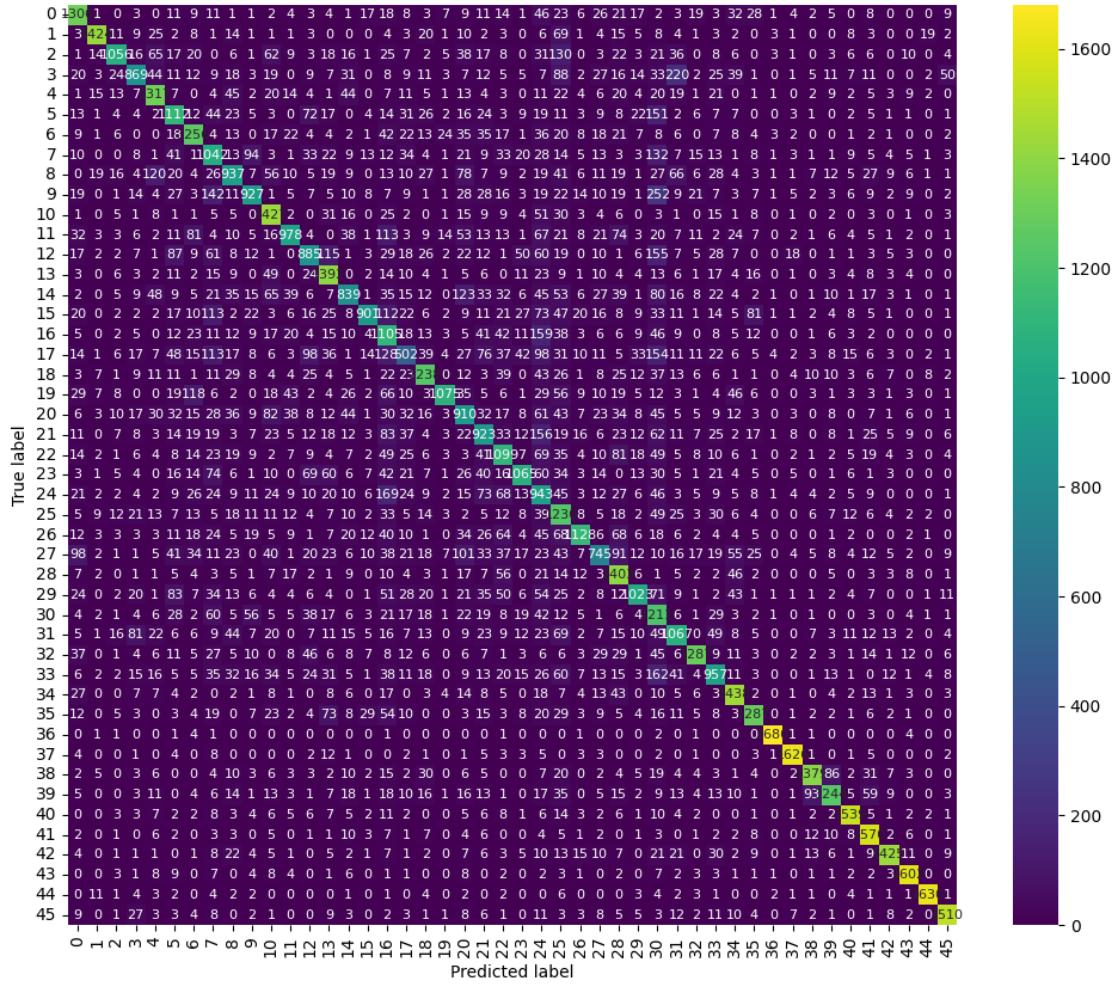
plt.figure(figsize=(12, 10))

```

```

sns.heatmap(svctrain_matrix, annot=True, fmt="d", cmap="viridis",
            annot_kws={"size":8})
plt.xlabel("Predicted label")
plt.ylabel("True label")
plt.show()

```



```

[32]: mse = mean_squared_error(target, svctrainpred)
print("MSE:", mse)

```

```

print(classification_report(target, svctrainpred))

```

MSE: 71.68058823529412

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.72	0.77	0.74	1700
1	0.92	0.84	0.88	1700
2	0.84	0.62	0.72	1700

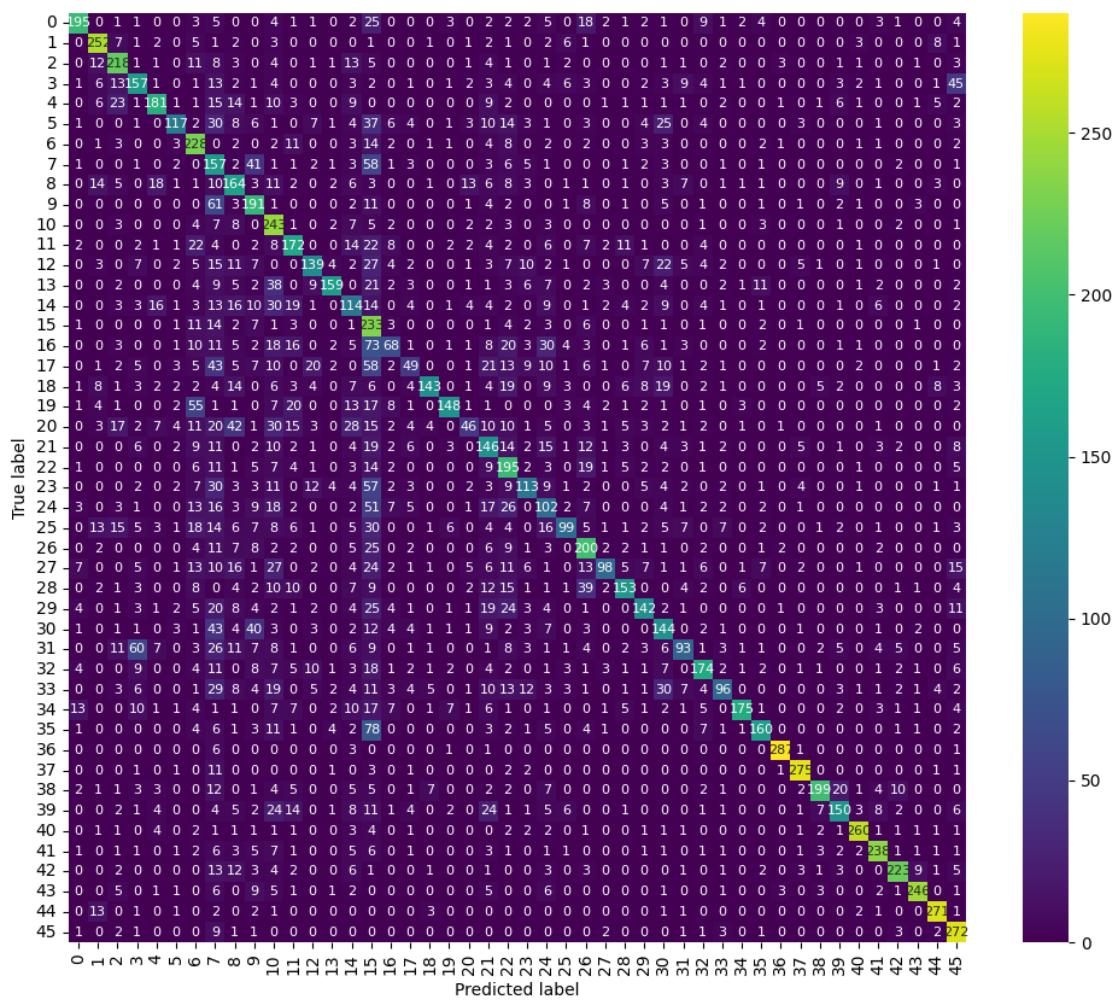
3	0.70	0.51	0.59	1700
4	0.72	0.77	0.75	1700
5	0.62	0.65	0.64	1700
6	0.70	0.74	0.72	1700
7	0.50	0.61	0.55	1700
8	0.63	0.55	0.59	1700
9	0.71	0.55	0.62	1700
10	0.67	0.84	0.74	1700
11	0.75	0.58	0.65	1700
12	0.60	0.52	0.56	1700
13	0.67	0.82	0.74	1700
14	0.65	0.49	0.56	1700
15	0.84	0.53	0.65	1700
16	0.43	0.65	0.52	1700
17	0.48	0.30	0.36	1700
18	0.74	0.73	0.73	1700
19	0.91	0.63	0.75	1700
20	0.50	0.54	0.52	1700
21	0.54	0.54	0.54	1700
22	0.60	0.65	0.62	1700
23	0.76	0.63	0.69	1700
24	0.37	0.55	0.45	1700
25	0.47	0.72	0.57	1700
26	0.83	0.66	0.74	1700
27	0.63	0.44	0.52	1700
28	0.62	0.83	0.71	1700
29	0.78	0.60	0.68	1700
30	0.38	0.71	0.50	1700
31	0.61	0.63	0.62	1700
32	0.86	0.76	0.81	1700
33	0.63	0.56	0.60	1700
34	0.76	0.85	0.80	1700
35	0.81	0.76	0.78	1700
36	0.98	0.99	0.99	1700
37	0.94	0.96	0.95	1700
38	0.88	0.81	0.84	1700
39	0.82	0.73	0.77	1700
40	0.90	0.91	0.90	1700
41	0.80	0.92	0.86	1700
42	0.92	0.84	0.88	1700
43	0.92	0.94	0.93	1700
44	0.97	0.96	0.97	1700
45	0.91	0.89	0.90	1700
accuracy			0.70	78200
macro avg	0.72	0.70	0.70	78200
weighted avg	0.72	0.70	0.70	78200

```
[33]: svctestpred = svcmodel.predict(test)
       print("svc Accuracy:", accuracy_score(testtarget, svctestpred))
```

svc Accuracy: 0.5998550724637681

```
[23]: svctest_matrix = confusion_matrix(testtarget, svctestpred)
```

```
plt.figure(figsize=(12, 10))
sns.heatmap(svctest_matrix, annot=True, fmt="d", cmap="viridis",
            annot_kws={"size":8})
plt.xlabel("Predicted label")
plt.ylabel("True label")
plt.show()
```



```
[34]: mse = mean_squared_error(testtarget, svctestpred)
      print("MSE:", mse)
```

```
print(classification_report(testtarget, svctestpred))
```

MSE: 104.91630434782608

	precision	recall	f1-score	support
0	0.64	0.68	0.66	300
1	0.79	0.72	0.75	300
2	0.74	0.55	0.63	300
3	0.55	0.42	0.48	300
4	0.64	0.71	0.67	300
5	0.55	0.56	0.56	300
6	0.58	0.66	0.61	300
7	0.40	0.49	0.44	300
8	0.53	0.46	0.49	300
9	0.63	0.48	0.54	300
10	0.59	0.74	0.66	300
11	0.69	0.52	0.59	300
12	0.54	0.44	0.48	300
13	0.58	0.71	0.64	300
14	0.54	0.42	0.47	300
15	0.69	0.43	0.53	300
16	0.34	0.55	0.42	300
17	0.37	0.23	0.28	300
18	0.64	0.59	0.61	300
19	0.82	0.55	0.66	300
20	0.40	0.44	0.42	300
21	0.43	0.41	0.42	300
22	0.55	0.60	0.57	300
23	0.69	0.48	0.57	300
24	0.26	0.41	0.32	300
25	0.35	0.54	0.42	300
26	0.70	0.54	0.61	300
27	0.52	0.34	0.41	300
28	0.55	0.64	0.59	300
29	0.67	0.55	0.60	300
30	0.31	0.61	0.41	300
31	0.49	0.46	0.48	300
32	0.75	0.62	0.68	300
33	0.52	0.46	0.48	300
34	0.66	0.78	0.71	300
35	0.69	0.61	0.65	300
36	0.93	0.95	0.94	300
37	0.84	0.92	0.88	300
38	0.78	0.75	0.77	300
39	0.71	0.59	0.64	300
40	0.82	0.85	0.83	300
41	0.73	0.79	0.76	300

42	0.79	0.75	0.77	300
43	0.84	0.87	0.86	300
44	0.85	0.90	0.88	300
45	0.79	0.83	0.81	300
accuracy			0.60	13800
macro avg	0.62	0.60	0.60	13800
weighted avg	0.62	0.60	0.60	13800

Result on SVM

sklearn.SVC is too slow, so i using SGDClassifier, which approximates a linear SVM using stochastic gradient descent — much faster and scalable to large datasets.

Hyperparameters: {'alpha': 0.0001, 'learning_rate': 'optimal', 'loss': 'hinge', 'penalty': 'l2'}

1. Overall Accuracy

- The model achieves ~70% accuracy on training and ~60% on test. • This gap shows some underfitting — the model learned general patterns but not enough to capture complex nonlinear handwriting variations. • MSE 71.68 (train) and 104.9 (test). this indicates misclassifications across different categories, not just near misses.

2. Class-wise Observations

- Some classes (like 36, 37, 44, 45) achieve very high precision and recall (0.9). • Others (like 16, 17, 24, 30) have F1-scores below 0.5, meaning these are harder to classify. • Some categories have high precision but low recall, meaning the classifier is conservative. • Others have low precision but higher recall, meaning the model is overgeneralizing to those labels.

The SGDClassifier (linear SVM) achieved ~70% training accuracy and ~60% test accuracy on the Devanagari Handwritten Character Dataset. This demonstrates that while the model learned some global pixel patterns, its linear nature limited performance on complex, nonlinear character shapes. Class-level performance varied significantly, with simpler or distinct characters achieving F1-scores above 0.9, while visually similar ones fell below 0.5.

Improve SVM

```
[6]: pipe = Pipeline([
    ('scaler', scaler),
    ('pca', PCA(n_components=0.95, whiten=True, random_state=42)),
    ('sgd', SGDClassifier(random_state=114514, max_iter=5000, tol=1e-4))
])
param_grid = {
    'sgd_loss': ['hinge', 'log_loss'],
    'sgd_alpha': [1e-4, 1e-3, 1e-2],
    'sgd_penalty': ['l2', 'l1', 'elasticnet'],
    'sgd_learning_rate': ['optimal', 'adaptive'],
    'sgd_eta0': [0.001, 0.01, 0.1]
}
grid = GridSearchCV(
```

```

    pipe,
    param_grid,
    cv=3,
    n_jobs=-1,
    verbose=2,
    scoring='accuracy'
)

grid = GridSearchCV(pipe, param_grid, cv=3, n_jobs=-1, verbose=2)
grid.fit(train[:10000], target[:10000])

print("Best params:", grid.best_params_)
print("Train acc:", grid.best_estimator_.score(train, target))
print("Test acc:", grid.best_estimator_.score(test, testtarget))
#Best params: {'sgd__alpha': 0.001, 'sgd__eta0': 0.1, 'sgd__learning_rate': 'adaptive', 'sgd__loss': 'log_loss', 'sgd__penalty': 'l1'}
#Train acc: 0.6658951406649616
#Test acc: 0.663695652173913

```

Fitting 3 folds for each of 108 candidates, totalling 324 fits

```

[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 3.7min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 15.9s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 22.9s

/opt/homebrew/anaconda3/envs/tf215/lib/python3.9/site-
packages/joblib/externals/loky/process_executor.py:782: UserWarning: A worker
stopped while some jobs were given to the executor. This can be caused by a too
short worker timeout or by a memory leak.

    warnings.warn(

```

```

[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 3.8min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 23.2s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 27.8s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 3.5min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 17.6s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 24.3s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 27.9s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,

```

```

sgd__loss=log_loss, sgd__penalty=l2; total time= 3.8min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 29.2s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 27.9s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 3.9min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 27.2s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 44.9s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 3.7min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 14.2s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 22.7s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 44.5s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 1.1min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 2.7min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 20.9s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 25.7s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 1.8min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 3.8min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 2.8min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 26.1s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 21.8s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 24.7s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.5min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 2.1min
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 2.1min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 42.5s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,

```

```

sgd__loss=hinge, sgd__penalty=l1; total time= 2.0min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 2.9min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 1.8min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 10.6s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 17.0s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 17.3s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 27.7s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 2.0min
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 9.4s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 15.0s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 17.1s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 45.5s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 3.0min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 4.6min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.8min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.1min
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 3.7min
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 3.3min
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 2.0min
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 8.8s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 18.8s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 3.1min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 4.5min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.9min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,

```

```

sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.1min
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 3.5min
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 3.8min
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 3.5min
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 1.1min
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 16.8s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 37.6s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.2min
Best params: {'sgd__alpha': 0.001, 'sgd__eta0': 0.1, 'sgd__learning_rate': 'adaptive', 'sgd__loss': 'log_loss', 'sgd__penalty': 'l1'}
Train acc: 0.6658951406649616
Test acc: 0.663695652173913
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 1.1min
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 7.5s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 10.4s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 28.6s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.1min
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 1.0min
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 7.0s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 7.4s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 11.4s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 11.8s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 27.5s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 43.4s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 2.5min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 14.1s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,

```

```

sgd__loss=hinge, sgd__penalty=l2; total time= 7.0s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 25.4s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 20.2s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 26.6s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 49.5s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 1.8min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 21.1s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 29.2s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 56.2s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 1.9min
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 18.7s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 29.7s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 39.9s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 1.8min
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 7.2s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 7.6s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 6.7s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 11.8s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 12.2s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 14.3s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 30.8s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.5min
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.4min
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 1.1min
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,

```

```

sgd__loss=log_loss, sgd__penalty=l1; total time= 37.8s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 34.1s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 1.2min
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 3.7min
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 8.5s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 11.6s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 33.9s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 45.1s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 2.0min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 2.9min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 1.9min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 10.8s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 21.0s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 16.4s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 26.6s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.5min
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.3min
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 2.0min
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 14.3s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 22.7s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.5min
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.5min
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 4.1min
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 7.8s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,

```

```

sgd__loss=log_loss, sgd__penalty=l2; total time= 10.1s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 28.4s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.2min
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 1.1min
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 5.8s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 10.0s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 10.8s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 34.4s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 9.2s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 15.2s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 17.0s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 44.7s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.5min
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.5min
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 3.6min
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 1.1min
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 14.6s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 38.1s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.2min
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 1.1min
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 6.3s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 11.1s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 27.8s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.3min
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,

```

```

sgd__loss=log_loss, sgd__penalty=l2; total time= 1.1min
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 11.8s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 12.2s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 32.8s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 1.1min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 2.4min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 25.7s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 28.2s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.5min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 2.1min
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.6min
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.4min
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 1.7min
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 11.1s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 14.8s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 22.5s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 57.6s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 2.0min
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 9.1s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 16.4s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 14.8s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 22.8s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.3min
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.2min
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,

```

```

sgd__loss=log_loss, sgd__penalty=l1; total time= 4.0min
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 12.4s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 33.1s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 1.7min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 3.4min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 1.2min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 2.7min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 30.5s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 51.9s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 1.9min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 21.6s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 30.5s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 59.1s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 1.9min
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 18.0s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 29.5s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.6min
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.2min
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 2.0min
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 15.1s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 15.2s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 29.3s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.3min
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.2min
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,

```

```

sgd__loss=log_loss, sgd__penalty=l1; total time= 4.3min
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 31.8s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.5min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 2.0min
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.5min
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.4min
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 3.8min
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 3.8min
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 3.9min
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 6.3s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 5.8s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 8.7s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 10.9s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 21.6s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 41.4s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 1.3min
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 2.7min
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 32.8s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.4min
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.2min
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.4min
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 2.6min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 42.8s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 2.6min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,

```

```

sgd__loss=hinge, sgd__penalty=l2; total time= 13.7s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 15.8s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 15.1s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 22.0s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 25.2s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 45.7s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 1.9min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 22.1s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 30.7s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 57.5s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 1.9min
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 44.2s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 57.7s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 2.1min
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 13.6s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 29.1s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 57.8s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 2.1min
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 10.2s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 15.0s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 32.7s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 26.1s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 1.3min
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 8.3s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,

```

```

sgd__loss=hinge, sgd__penalty=l1; total time= 7.6s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 12.5s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 38.0s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 33.4s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 1.2min
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.4min
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 2.8min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 1.7min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 3.6min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.9min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 2.8min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 1.9min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 10.8s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 21.6s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 16.8s
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 27.3s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 2.0min
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 9.5s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 15.7s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 17.5s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 18.2s
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 29.6s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.5min
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.5min
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,

```

```

sgd__loss=log_loss, sgd__penalty=12; total time= 2.0min
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 16.3s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 32.0s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 25.5s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 1.2min
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 8.1s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l2; total time= 6.2s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 7.5s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 12.3s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 37.8s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 33.8s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 1.2min
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.2min
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 3.2min
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 58.3s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 2.1min
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 13.2s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 14.0s
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 22.7s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.6min
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.5min
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 4.2min
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 10.0s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 21.3s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,

```

```

sgd__loss=hinge, sgd__penalty=12; total time= 26.3s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 1.2min
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 3.6min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 3.0min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 4.5min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.8min
[CV] END sgd__alpha=0.0001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 3.2min
[CV] END sgd__alpha=0.001, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 3.4min
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 2.0min
[CV] END sgd__alpha=0.001, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l1; total time= 30.6s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 57.4s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 2.0min
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 15.6s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 12.6s
[CV] END sgd__alpha=0.001, sgd__eta0=0.1, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 29.1s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 37.8s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 1.2min
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=l1; total time= 7.9s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 12.0s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=l2; total time= 15.1s
[CV] END sgd__alpha=0.01, sgd__eta0=0.001, sgd__learning_rate=adaptive,
sgd__loss=log_loss, sgd__penalty=elasticnet; total time= 36.9s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l1; total time= 1.2min
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l2; total time= 1.1min
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 10.2s
[CV] END sgd__alpha=0.01, sgd__eta0=0.01, sgd__learning_rate=adaptive,

```

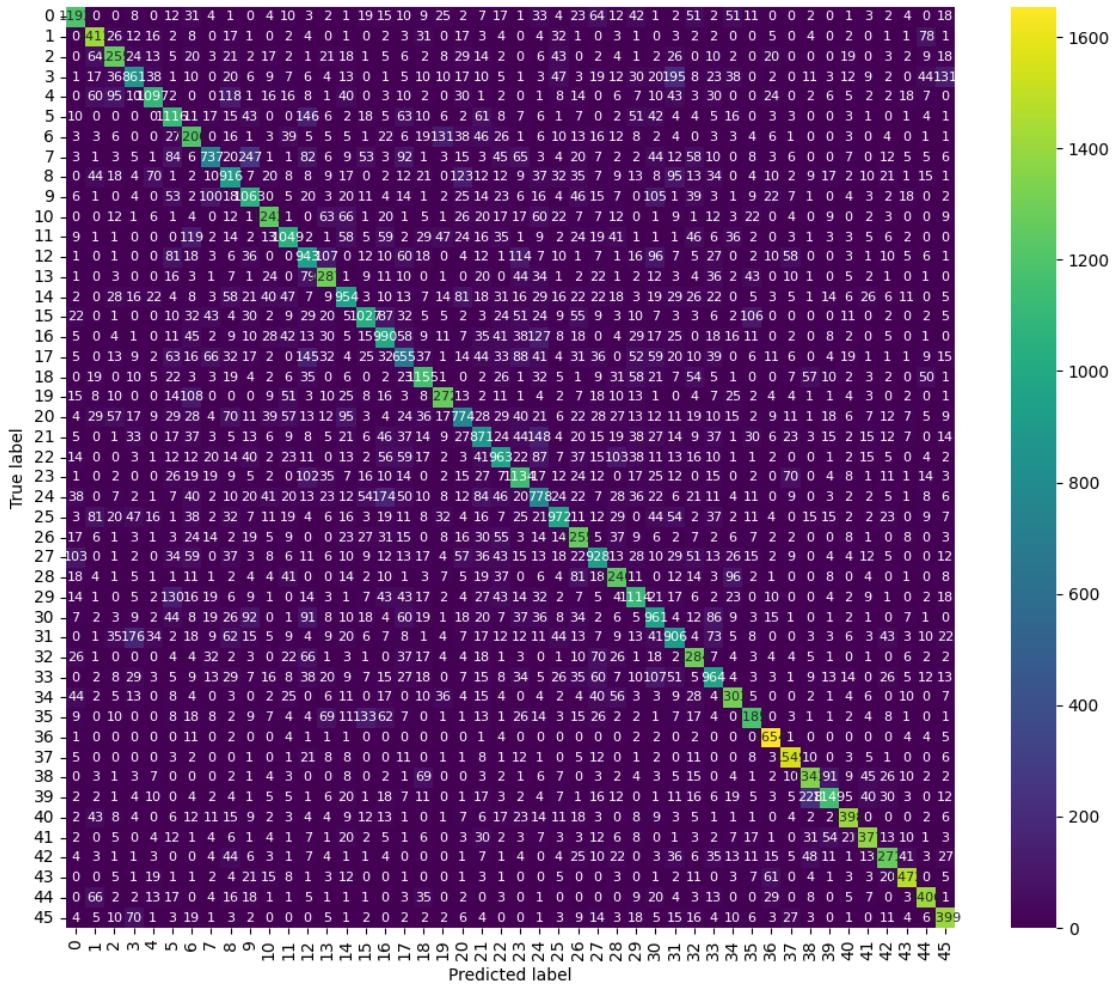
```
sgd__loss=log_loss, sgd__penalty=l1; total time= 27.8s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=l2; total time= 40.1s
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=hinge, sgd__penalty=elasticnet; total time= 1.0min
[CV] END sgd__alpha=0.01, sgd__eta0=0.1, sgd__learning_rate=optimal,
sgd__loss=log_loss, sgd__penalty=l1; total time= 3.6min
```

```
[23]: bestpipe_model = grid.best_estimator_
bestpipe_trainpred = bestpipe_model.predict(train)
print("svc Accuracy:", accuracy_score(target, bestpipe_trainpred))
```

```
svc Accuracy: 0.6658951406649616
```

```
[24]: svcpipetrain_matrix = confusion_matrix(target, bestpipe_trainpred)

plt.figure(figsize=(12, 10))
sns.heatmap(svcpipetrain_matrix, annot=True, fmt="d", cmap="viridis",
            annot_kws={"size":8})
plt.xlabel("Predicted label")
plt.ylabel("True label")
plt.show()
```



```
[25]: mse = mean_squared_error(target, bestpipe_trainpred)
print("MSE:", mse)

print(classification_report(target, bestpipe_trainpred))
```

MSE: 89.00122762148338

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.75	0.70	0.72	1700
1	0.75	0.83	0.79	1700
2	0.74	0.74	0.74	1700
3	0.62	0.51	0.56	1700
4	0.79	0.65	0.71	1700
5	0.59	0.66	0.62	1700
6	0.59	0.71	0.64	1700
7	0.62	0.43	0.51	1700
8	0.53	0.54	0.53	1700

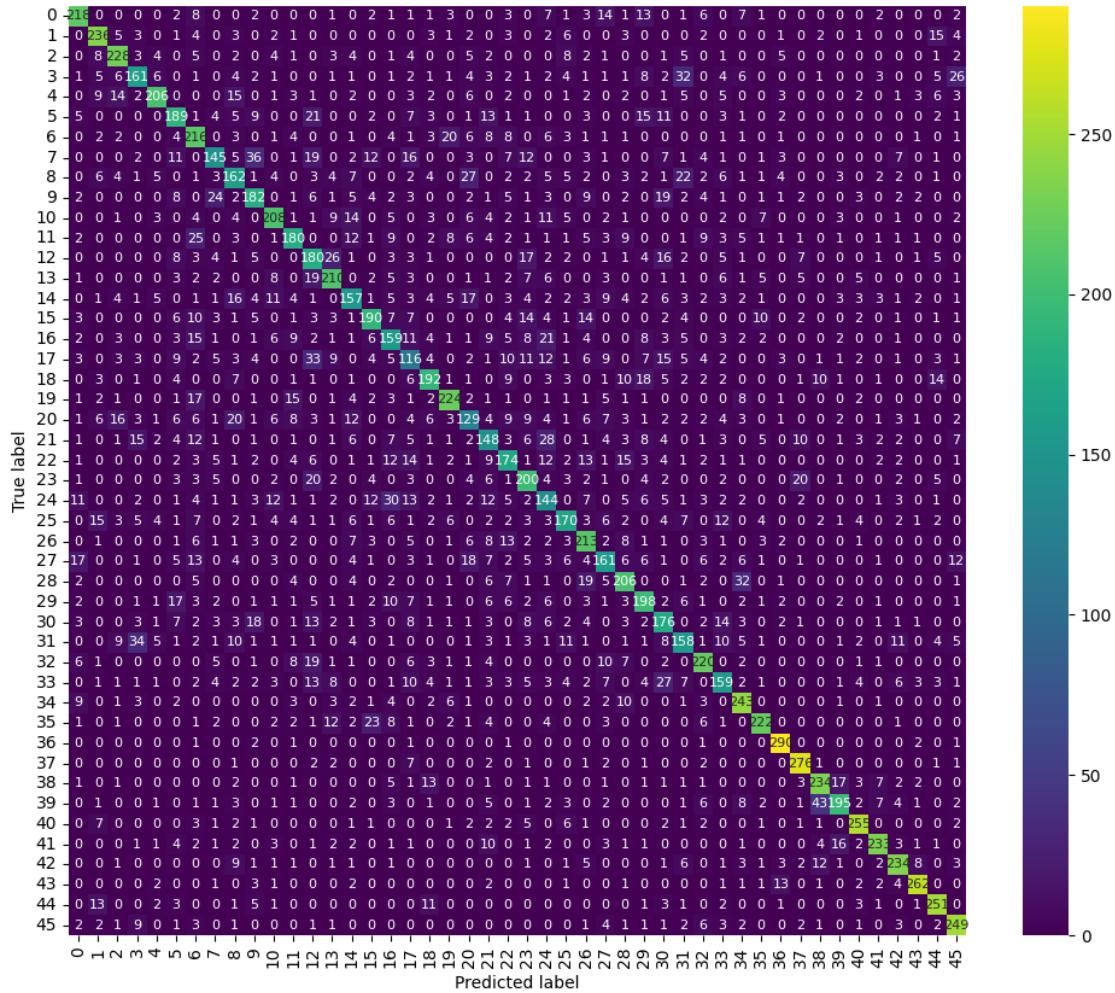
9	0.59	0.63	0.61	1700
10	0.77	0.73	0.75	1700
11	0.67	0.62	0.64	1700
12	0.48	0.55	0.51	1700
13	0.69	0.76	0.72	1700
14	0.59	0.56	0.58	1700
15	0.68	0.60	0.64	1700
16	0.53	0.58	0.56	1700
17	0.43	0.39	0.41	1700
18	0.68	0.68	0.68	1700
19	0.76	0.75	0.75	1700
20	0.53	0.46	0.49	1700
21	0.51	0.51	0.51	1700
22	0.57	0.57	0.57	1700
23	0.59	0.67	0.62	1700
24	0.45	0.46	0.45	1700
25	0.67	0.57	0.62	1700
26	0.63	0.74	0.68	1700
27	0.58	0.55	0.56	1700
28	0.67	0.73	0.70	1700
29	0.65	0.66	0.65	1700
30	0.53	0.57	0.55	1700
31	0.53	0.53	0.53	1700
32	0.67	0.76	0.71	1700
33	0.59	0.57	0.58	1700
34	0.74	0.77	0.75	1700
35	0.75	0.70	0.72	1700
36	0.85	0.97	0.91	1700
37	0.84	0.91	0.87	1700
38	0.74	0.79	0.77	1700
39	0.78	0.68	0.72	1700
40	0.86	0.82	0.84	1700
41	0.84	0.81	0.82	1700
42	0.79	0.75	0.77	1700
43	0.88	0.87	0.87	1700
44	0.82	0.83	0.82	1700
45	0.77	0.82	0.80	1700
accuracy			0.67	78200
macro avg	0.67	0.67	0.66	78200
weighted avg	0.67	0.67	0.66	78200

```
[26]: bestpipe_testpred = bestpipe_model.predict(test)
print("svc Accuracy:", accuracy_score(testtarget, bestpipe_testpred))
```

svc Accuracy: 0.663695652173913

```
[27]: svcpiptest_matrix = confusion_matrix(testtarget, bestpipe_testpred)

plt.figure(figsize=(12, 10))
sns.heatmap(svcpiptest_matrix, annot=True, fmt="d", cmap="viridis",
            annot_kws={"size":8})
plt.xlabel("Predicted label")
plt.ylabel("True label")
plt.show()
```



```
[28]: mse = mean_squared_error(testtarget, bestpipe_testpred)
print("MSE:", mse)

print(classification_report(testtarget, bestpipe_testpred))
```

MSE: 89.27210144927537

precision	recall	f1-score	support
-----------	--------	----------	---------

0	0.74	0.73	0.73	300
1	0.74	0.79	0.76	300
2	0.75	0.76	0.75	300
3	0.63	0.54	0.58	300
4	0.82	0.69	0.75	300
5	0.62	0.63	0.62	300
6	0.55	0.72	0.62	300
7	0.64	0.48	0.55	300
8	0.54	0.54	0.54	300
9	0.61	0.61	0.61	300
10	0.72	0.69	0.71	300
11	0.68	0.60	0.64	300
12	0.48	0.60	0.53	300
13	0.68	0.70	0.69	300
14	0.58	0.52	0.55	300
15	0.68	0.63	0.66	300
16	0.53	0.53	0.53	300
17	0.42	0.39	0.40	300
18	0.69	0.64	0.66	300
19	0.77	0.75	0.76	300
20	0.49	0.43	0.46	300
21	0.50	0.49	0.50	300
22	0.58	0.58	0.58	300
23	0.59	0.67	0.63	300
24	0.46	0.48	0.47	300
25	0.66	0.57	0.61	300
26	0.61	0.71	0.66	300
27	0.59	0.54	0.56	300
28	0.66	0.69	0.67	300
29	0.62	0.66	0.64	300
30	0.52	0.59	0.55	300
31	0.55	0.53	0.54	300
32	0.72	0.73	0.73	300
33	0.60	0.53	0.56	300
34	0.69	0.81	0.75	300
35	0.81	0.74	0.77	300
36	0.85	0.97	0.91	300
37	0.82	0.92	0.87	300
38	0.75	0.78	0.76	300
39	0.77	0.65	0.71	300
40	0.84	0.85	0.85	300
41	0.84	0.78	0.81	300
42	0.78	0.78	0.78	300
43	0.89	0.87	0.88	300
44	0.77	0.84	0.80	300
45	0.75	0.83	0.79	300

accuracy			0.66	13800
macro avg	0.66	0.66	0.66	13800
weighted avg	0.66	0.66	0.66	13800

Result on SVM + PCA

1. Overall • PCA reduced noise and improved generalization slightly (compare: SVM raw test 0.60 → SVM+PCA 0.66). • The smaller gap between train and test accuracy (0.66 vs 0.66) indicates less overfitting and better stability. • MSE dropped significantly (104.9 → 89.3), confirming fewer large misclassifications. • The model is now balanced — neither overfitting nor heavily underfitting.
2. Class-wise Behavior • Classes with clear geometric structure (like digits 36, 37, 44, 45) achieved F1 0.9+, showing strong linear separability after PCA compression. • Ambiguous or visually similar classes (16, 17, 24, 30) still struggle (F1 0.4–0.5). • PCA successfully improved generalization by focusing on the most informative features (top ~95% variance) while discarding noise.

PCA improved efficiency and generalization for the linear SVM without hurting performance.

Applying PCA to SVM features improved test accuracy by approximately 6%, reduced overfitting, and made training more efficient. The PCA-transformed model retained essential shape information while filtering noise, resulting in better generalization to unseen handwritten characters.

over all insight

- Random Forest captures complex nonlinear structures (curves, crossings, strokes) that SVMs can't.
- SVM models focus on hyperplanes; they can only separate classes linearly in the PCA-reduced space.
- PCA helps SVM reach moderate accuracy, but its ceiling is still limited (~65–70%).
- Random Forest's perfect training accuracy (1.0) and strong test accuracy (0.92) show that it's superior for this task.

Conclusion

In this project, multiple supervised learning models were applied to the Devanagari Handwritten Character Dataset to evaluate their ability to classify 46 different handwritten characters. The models included a Support Vector Machine (SVM) implemented via SGDClassifier, both with and without Principal Component Analysis (PCA), and a Random Forest classifier as a nonlinear ensemble baseline.

The baseline linear SVM trained on raw pixel features achieved approximately 70% training accuracy and 60% test accuracy, indicating mild overfitting. After applying PCA to retain 95% of the data variance, the SVM + PCA model achieved 66.6% training accuracy and 66.4% test accuracy, showing improved generalization and reduced error (MSE decreased from 104.9 to 89.3). PCA successfully filtered out noise and redundant information, enabling the model to focus on the most informative feature components. Although accuracy improved moderately, certain visually similar characters (e.g., tha, dha) remained challenging for the linear classifier due to the complex, nonlinear structure of handwritten strokes.

In comparison, the Random Forest model achieved a test accuracy of 92% and an almost perfect training accuracy of 100%, demonstrating its superior ability to capture nonlinear relationships and spatial patterns among pixels. While Random Forest offers the highest accuracy, the SVM +

PCA model provides a computationally efficient alternative with acceptable generalization. Overall, ensemble-based methods outperform linear models for handwritten character recognition, but dimensionality reduction techniques like PCA remain valuable for improving efficiency and robustness in high dimensional datasets.

[]: