1. Dataset Description

Wine Dataset

- Source: UCI Machine Learning Repository.
- Features: 13 continuous attributes.
- Target: 3 wine classes.
- Number of instances: 178 samples.

Handwritten Digit Dataset

- Source: UCI Machine Learning Repository.
- Features: 64 attributes (8×8 pixel grayscale image values).
- Target: Digits 0–9.
- Number of instances: 5,620 samples.

---

2. Objective of the Report

The purpose of this study is to evaluate the performance of multiple machine learning classifiers on two datasets.

1. **Classification Models**
   - Support Vector Machine (Linear, Polynomial, RBF, Sigmoid kernels)
   - Multi-Layer Perceptron (MLP)
   - Random Forest
2. **Evaluation Metrics**
   - Accuracy, Precision, Recall, F1-score
   - Confusion Matrix (heatmap)
   - ROC Curve and AUC
3. **Experiments Conducted**
   - Different train-test splits: **50:50, 60:40, 70:30, 80:20**.
   - **Parameter tuning** using GridSearchCV.
   - **Dimensionality reduction** using Principal Component Analysis.

The goal was to achieve **≥90% accuracy** and to compare performance across classifiers and settings.

---

3. Methodology

Helper Functions

- `evaluate_model()` → Calculates metrics and stores results in DataFrame.

```
def evaluate_model(y_true, y_pred, model_name, split_ratio,
results_df, title):
    acc = accuracy_score(y_true, y_pred)
    prec = precision_score(y_true, y_pred, average='weighted')
    rec = recall_score(y_true, y_pred, average='weighted')
    f1 = f1_score(y_true, y_pred, average='weighted')

    results_df.loc[len(results_df)] = [
        model_name, split_ratio, acc, prec, rec, f1
    ]

    print("\nEvaluation Results:")
    print(results_df.tail(1))

    return results_df
```

- `plot_confusion()` → Confusion matrix heatmap.

```
def plot_confusion(y_true, y_pred, title):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(6,5))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Reds')
    plt.title(f'Confusion Matrix - {title}')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.savefig(os.path.join(folder_path, f'Confusion_Matrix -
{title}'))
    plt.show()
    plt.close()
```

- `plot_roc_curve()` → ROC curve with AUC.

```
def plot_roc_curve(model, X_test, y_test, title, n_classes):
   y_score = model.decision_function(X_test) if hasattr(model, "decision_function") else model.predict_proba(X_test)
   y_test_bin = label_binarize(y_test, classes=np.arange(n_classes))

   plt.figure(figsize=(6,5))
   for i in range(n_classes):
       fpr, tpr, _ = roc_curve(y_test_bin[:, i], y_score[:, i])
       plt.plot(fpr, tpr, label=f'Class {i} (AUC = {auc(fpr, tpr):.2f})')
   plt.plot([0,1], [0,1], 'k--')
   plt.xlabel('False Positive Rate')
   plt.ylabel('True Positive Rate')
   plt.title(f'ROC Curve - {title}')
   plt.legend()
   plt.savefig(os.path.join(folder_path, f'ROC Curve - {title}'))
   plt.show()
   plt.close()
```

- `train_and_evaluate()` → Main driver to train, tune, evaluate classifiers on different splits.

```
def train_and_evaluate(X, y, dataset_name):
    results = pd.DataFrame(columns=['Model', 'Split', 'Accuracy',
'Precision', 'Recall', 'F1-score'])
    splits = [0.5, 0.4, 0.3, 0.2]

    classifiers = {
        'SVM-linear': SVC(kernel='linear', probability=True),
        'SVM-poly': SVC(kernel='poly', probability=True),
        'SVM-rbf': SVC(kernel='rbf', probability=True),
        'SVM-sigmoid': SVC(kernel='sigmoid', probability=True),
        'MLP': MLPClassifier(max_iter=500, random_state=42),
        'RandomForest': RandomForestClassifier(random_state=42)
    }

    param_grids = {
        'SVM-linear': {
            'C': [0.1, 1, 10]
        },
        'SVM-poly': {
            'C': [0.1, 1, 10],
            'degree': [2, 3, 4],
            'gamma': ['scale', 'auto']
        },
        'SVM-rbf': {
            'C': [0.1, 1, 10],
            'gamma': ['scale', 'auto', 0.01, 0.1]
        },
        'SVM-sigmoid': {
            'C': [0.1, 1, 10],
            'gamma': ['scale', 'auto']
        },
        'MLP': {
            'hidden_layer_sizes': [(50,), (100,), (100, 50)],
            'activation': ['tanh', 'relu'],
            'solver': ['sgd', 'adam'],
            'learning_rate_init': [0.001, 0.01],
            'momentum': [0.8, 0.9],
            'max_iter': [300, 500]
        },
```

```
        'RandomForest': {
            'n_estimators': [100, 200],
            'max_depth': [None, 10, 20],
            'min_samples_split': [2, 5],
            'min_samples_leaf': [1, 2]
        }
    }


    for split in splits:
        X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size=split, random_state=42, stratify=y
        )
        scaler = StandardScaler()
        X_train = scaler.fit_transform(X_train)
        X_test = scaler.transform(X_test)

        for name, clf in classifiers.items():
            split_ratio = f'{int((1 - split) * 100)}:{int(split * 100)}'
            title = f'{name}-{dataset_name} ({split_ratio})'
            print(f"{name} train-test split: {split_ratio}")

            # Train default model
            clf.fit(X_train, y_train)
            y_pred = clf.predict(X_test)
            print(f"[Default] {name} accuracy: {accuracy_score(y_test, y_pred):.4f}")

            # Tune model with GridSearchCV
            grid = GridSearchCV(clf, param_grids[name], cv=3, scoring='accuracy', n_jobs=-1)
            grid.fit(X_train, y_train)
            best_model = grid.best_estimator_
            y_pred_tuned = best_model.predict(X_test)

            print(f"[Tuned] {name} accuracy: {accuracy_score(y_test, y_pred_tuned):.4f}")
            print(f"[Tuned] {name} best params: {grid.best_params_}")
```

```
            # Evaluate
            results = evaluate_model(
                y_test, y_pred_tuned, f'{name}-{dataset_name}',
                split_ratio, results, title
            )
            plot_confusion(y_test, y_pred_tuned, title)
            plot_roc_curve(clf, X_test, y_test, title,
len(np.unique(y))

    results.to_csv(os.path.join(folder_path, f'{dataset_name}.csv'),
index=False)
    return results
```

- `train_and_evaluate_with_pca()` → PCA wrapper function.

```
def train_and_evaluate_with_pca(X, y, dataset_name,
n_components=0.95):
    scaler = StandardScaler()
    X = scaler.fit_transform(X)
    pca = PCA(n_components=n_components, random_state=42)
    X_pca = pca.fit_transform(X)

    print(f"[{dataset_name}-PCA] Reduced dimensionality: {X.shape[1]}
-> {X_pca.shape[1]}")

    return train_and_evaluate(X_pca, y, f"{dataset_name}-PCA")
```

---

4. Experimental Results

Since 48 outputs (confusion matrices and ROC curves, for 6 classifier modes using 4 test-splits) per dataset are impractical to display, I am summarizing results as follows:

- For each **classifier**, the **best train-test split** outcome (highest Accuracy) is presented.
- Both **normal evaluation** and **PCA-reduced evaluation** are included.
- Complete results are available in github (`classification_results.csv`, 98 rows).

4.1 Wine Dataset

```
wine = load_wine()
results_wine = train_and_evaluate(wine.data, wine.target, "Wine")
results_wine_pca = train_and_evaluate_with_pca(wine.data, wine.target, "Wine")
```

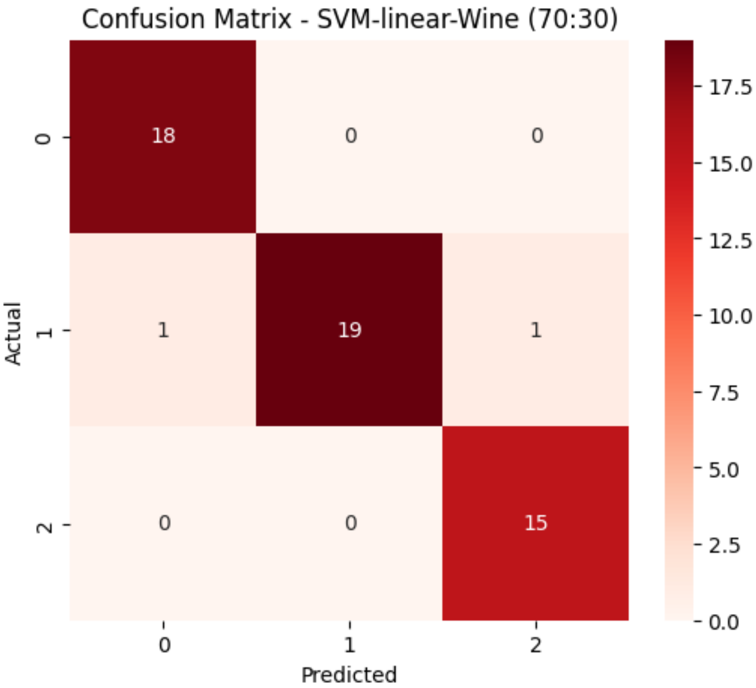
Best Results:

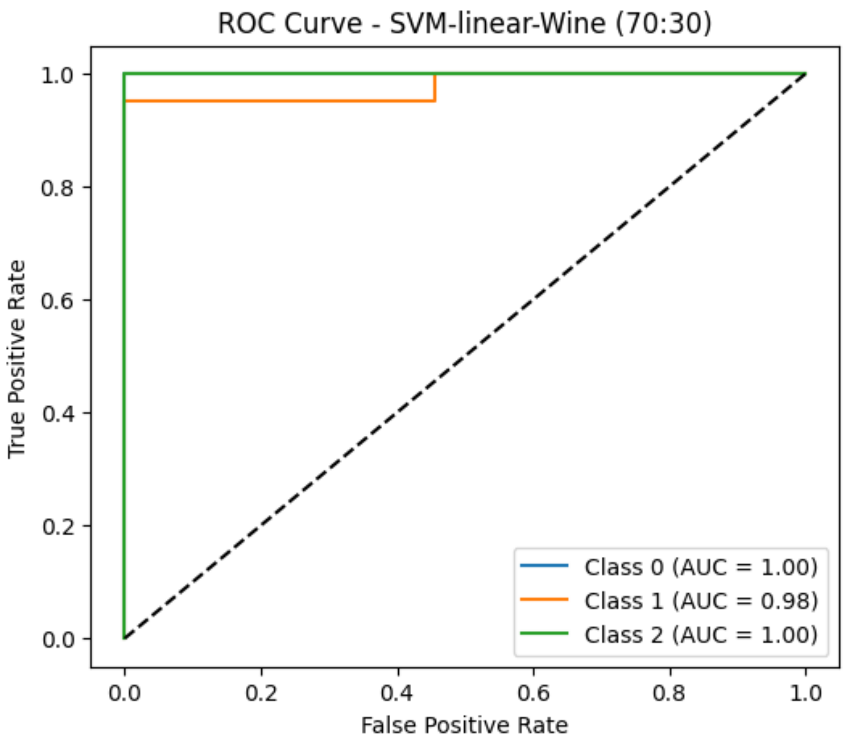
Model	PCA	Split	Accuracy	Precision	Recall	F1-score
SVM-linear	No	70:30	0.9630	0.9651	0.9630	0.9626
SVM-poly	No	80:20	0.9167	0.9225	0.9167	0.9156
SVM-rbf	No	70:30	0.9815	0.9823	0.9815	0.9814
SVM-sigmoid	No	60:40	0.9722	0.9735	0.9722	0.9720
MLP	No	70:30	0.9815	0.9825	0.9815	0.9815
RandomForest	No	80:20	1.0000	1.0000	1.0000	1.0000
SVM-linear	Yes	70:30	0.9630	0.9673	0.9630	0.9632
SVM-poly	Yes	80:20	0.9722	0.9744	0.9722	0.9723
SVM-rbf	Yes	70:30	0.9630	0.9630	0.9630	0.9630
SVM-sigmoid	Yes	60:40	0.9861	0.9868	0.9861	0.9862
MLP	Yes	70:30	0.9815	0.9826	0.9815	0.9816
RandomForest	Yes	80:20	0.9444	0.9444	0.9444	0.9444

SVM-linear train-test split: 70:30
[Default] SVM-linear accuracy: 0.9630
[Tuned] SVM-linear accuracy: 0.9630
[Tuned] SVM-linear best params: {'C': 0.1}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
0	SVM-linear-Wine	70:30	0.962963	0.965095	0.962963	0.962586

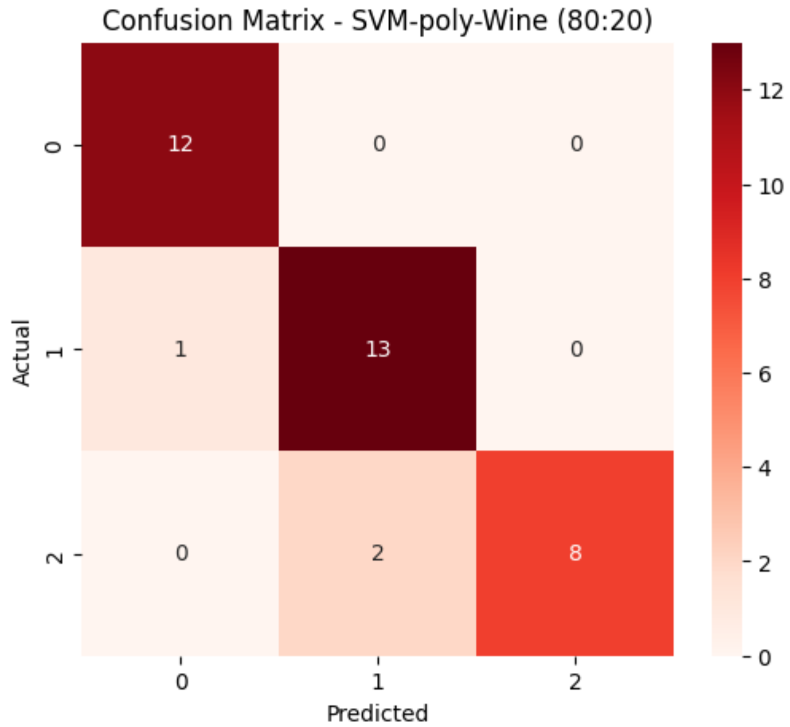


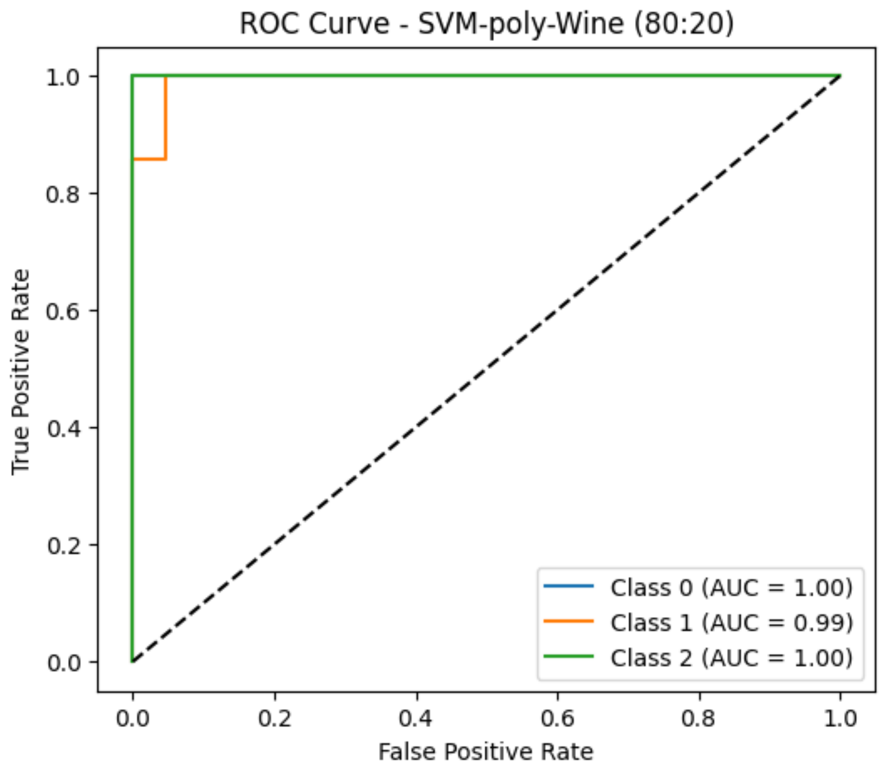


SVM-poly train-test split: 80:20
[Default] SVM-poly accuracy: 0.9444
[Tuned] SVM-poly accuracy: 0.9167
[Tuned] SVM-poly best params: {'C': 10, 'degree': 3, 'gamma': 'scale'}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
0	SVM-poly-Wine	80:20	0.916667	0.922507	0.916667	0.915573

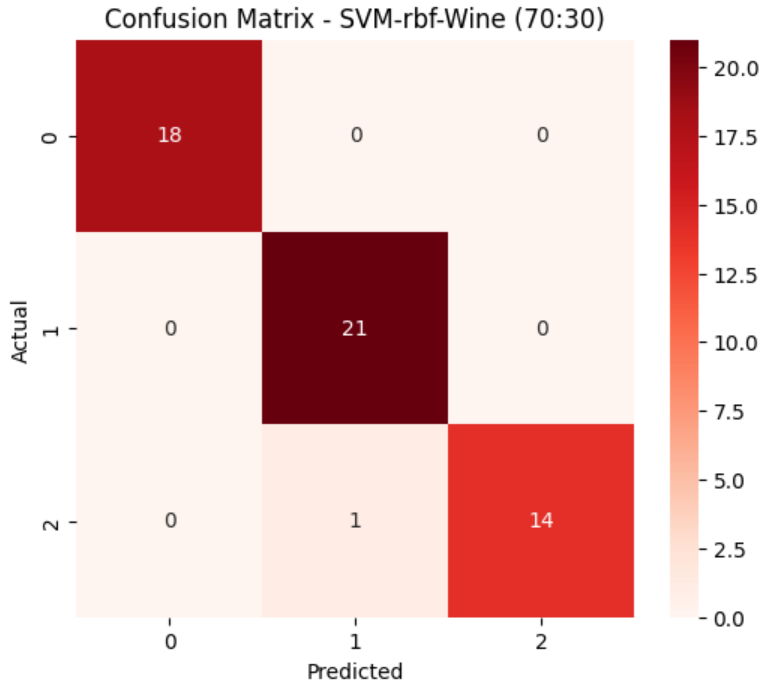


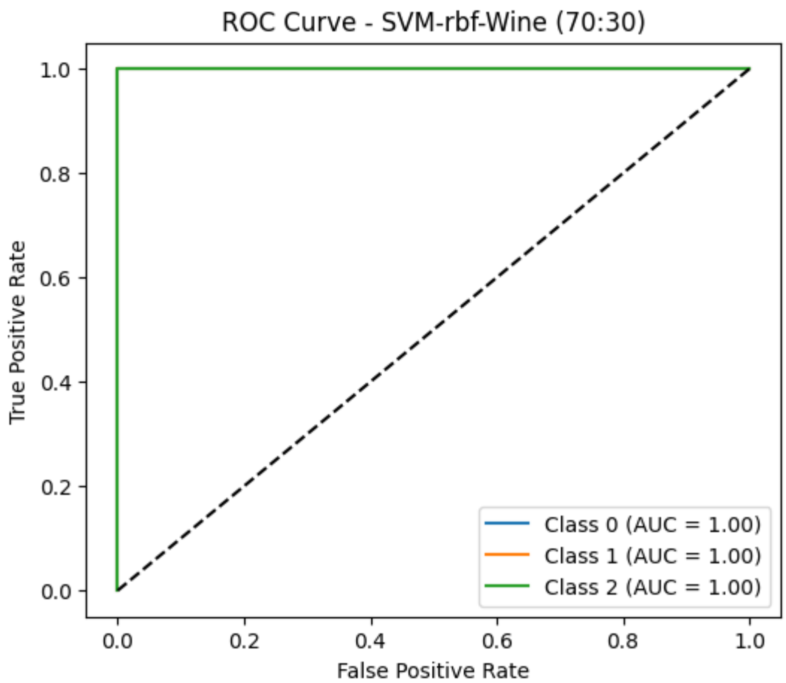


SVM-rbf train-test split: 70:30
[Default] SVM-rbf accuracy: 0.9815
[Tuned] SVM-rbf accuracy: 0.9815
[Tuned] SVM-rbf best params: {'C': 1, 'gamma': 'scale'}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
0	SVM-rbf-Wine	70:30	0.981481	0.982323	0.981481	0.981378

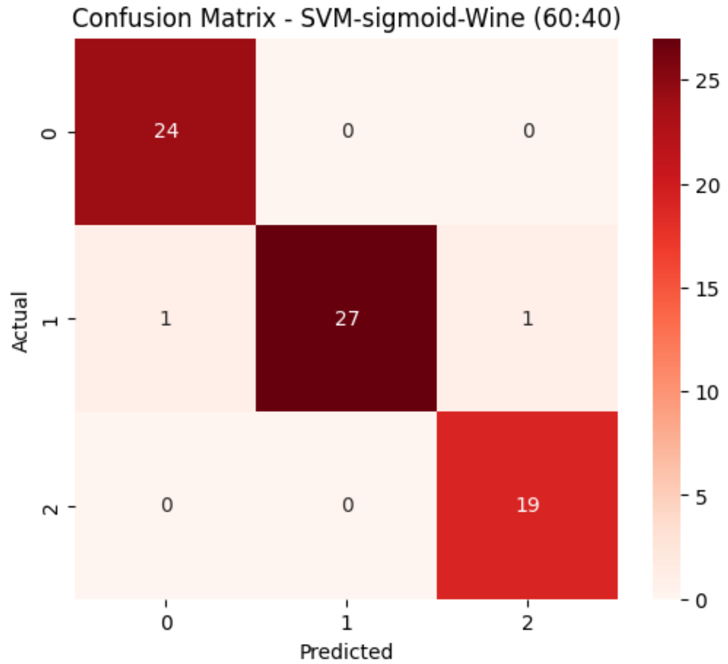


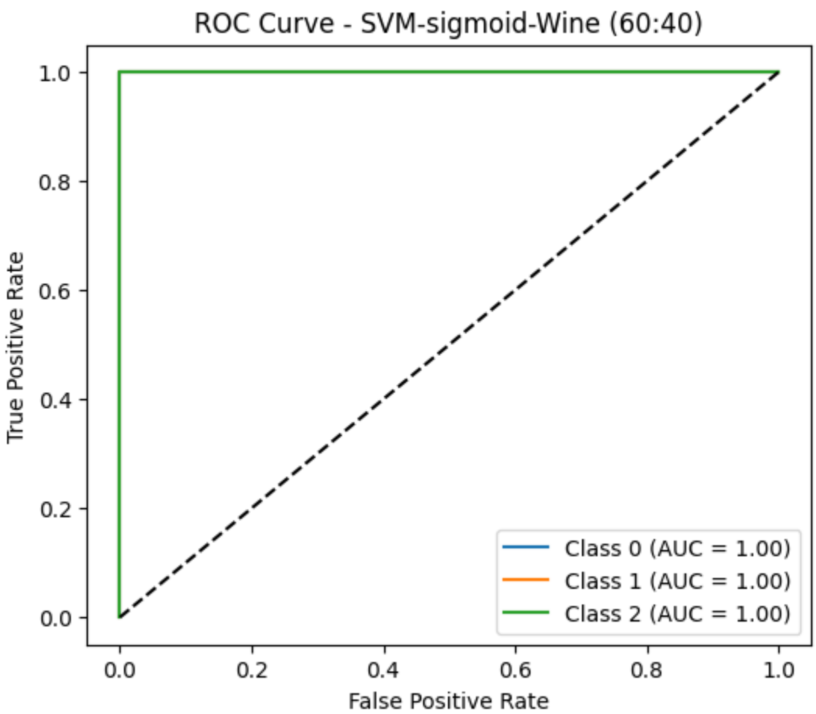


SVM-sigmoid train-test split: 60:40
[Default] SVM-sigmoid accuracy: 1.0000
[Tuned] SVM-sigmoid accuracy: 0.9722
[Tuned] SVM-sigmoid best params: {'C': 0.1, 'gamma': 'scale'}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
0	SVM-sigmoid-Wine	60:40	0.972222	0.973472	0.972222	0.972046

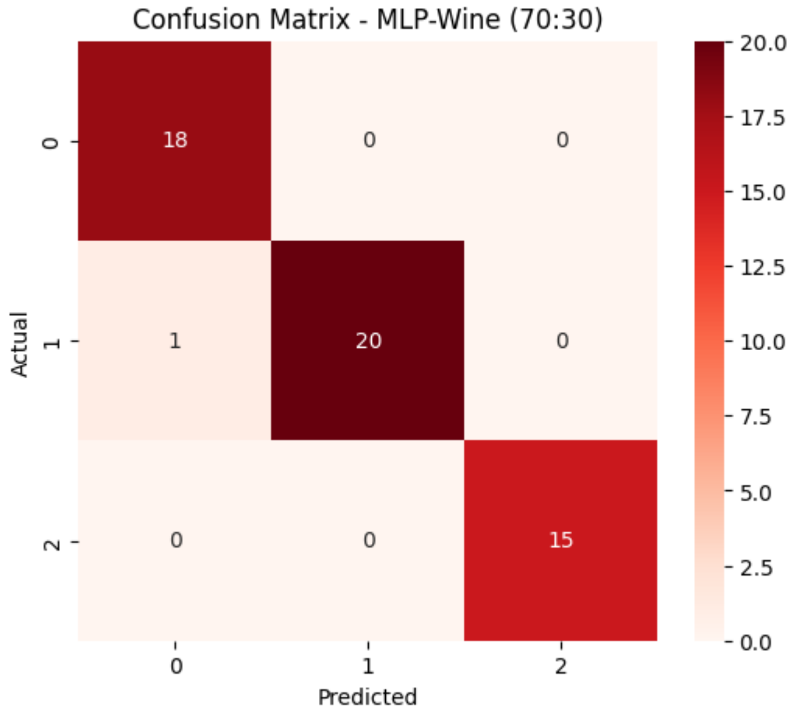


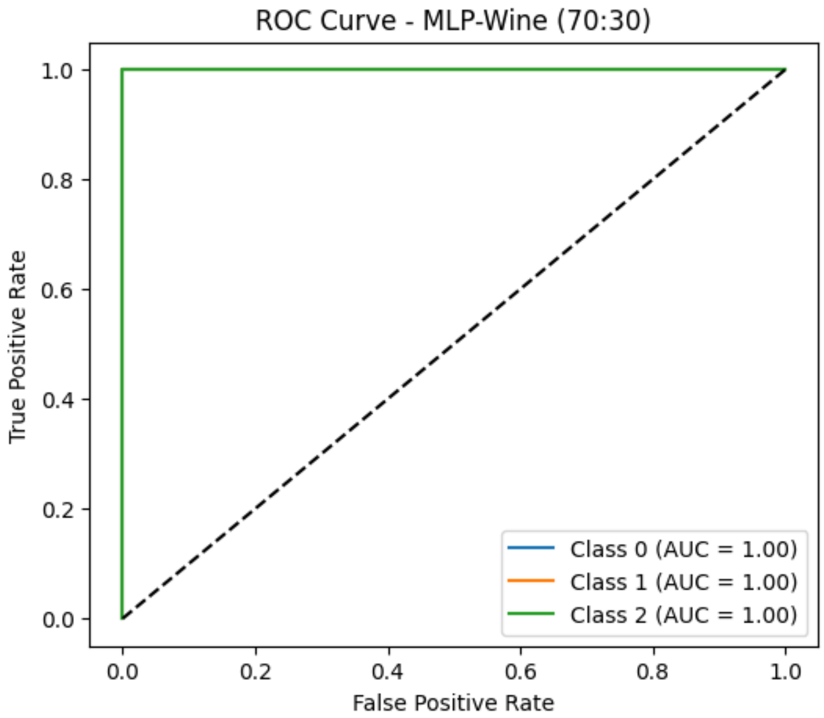


MLP train-test split: 70:30
[Default] MLP accuracy: 1.0000
[Tuned] MLP accuracy: 0.9815
[Tuned] MLP best params: {'activation': 'tanh', 'hidden_layer_sizes': (100,), 'learning_rate_init': 0.001, 'max_iter': 500, 'momentum': 0.9, 'solver': 'sgd'}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
1	MLP-Wine	70:30	0.981481	0.982456	0.981481	0.981506

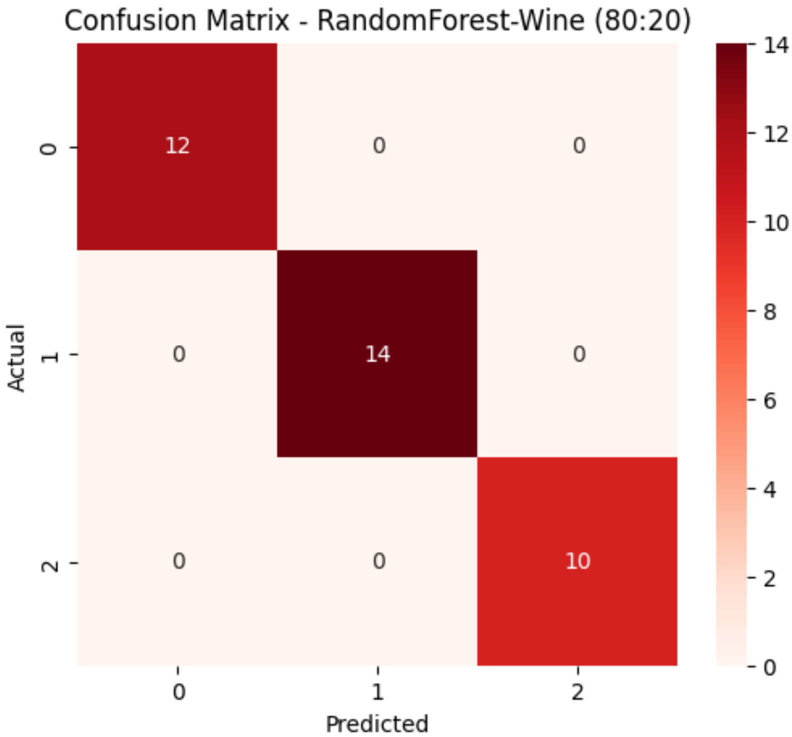


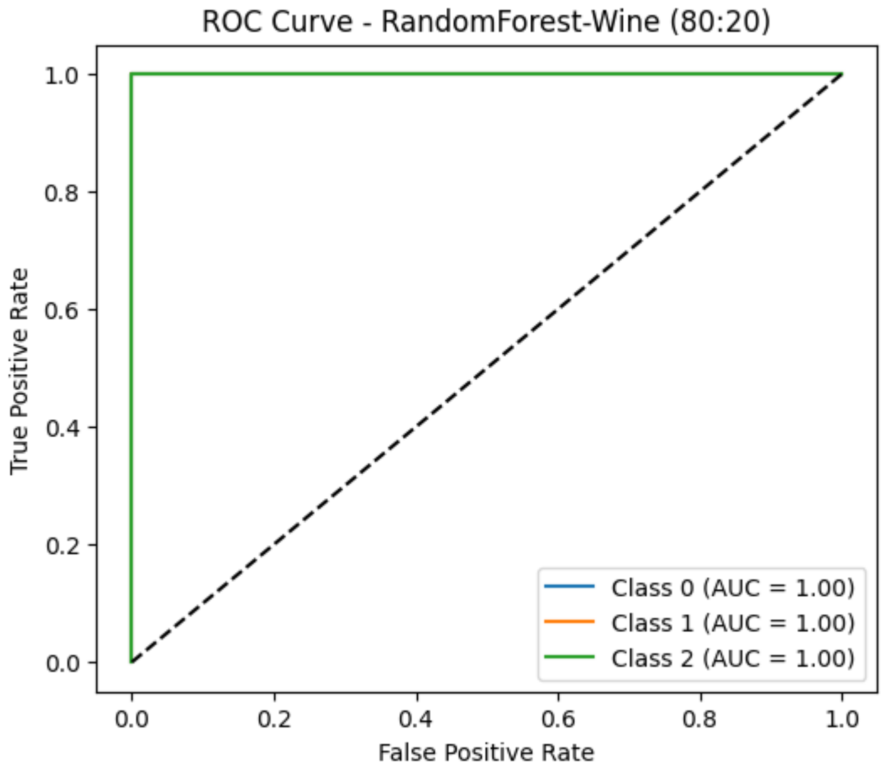


RandomForest train-test split: 80:20
[Default] RandomForest accuracy: 1.0000
[Tuned] RandomForest accuracy: 1.0000
[Tuned] RandomForest best params: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
0	RandomForest-Wine	80:20	1.0	1.0	1.0	1.0

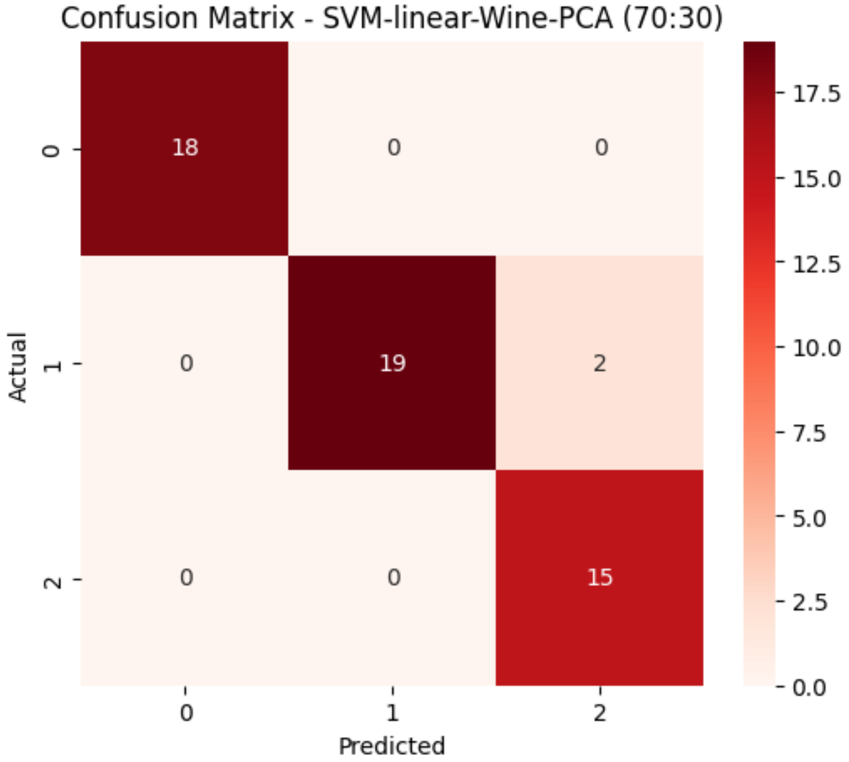


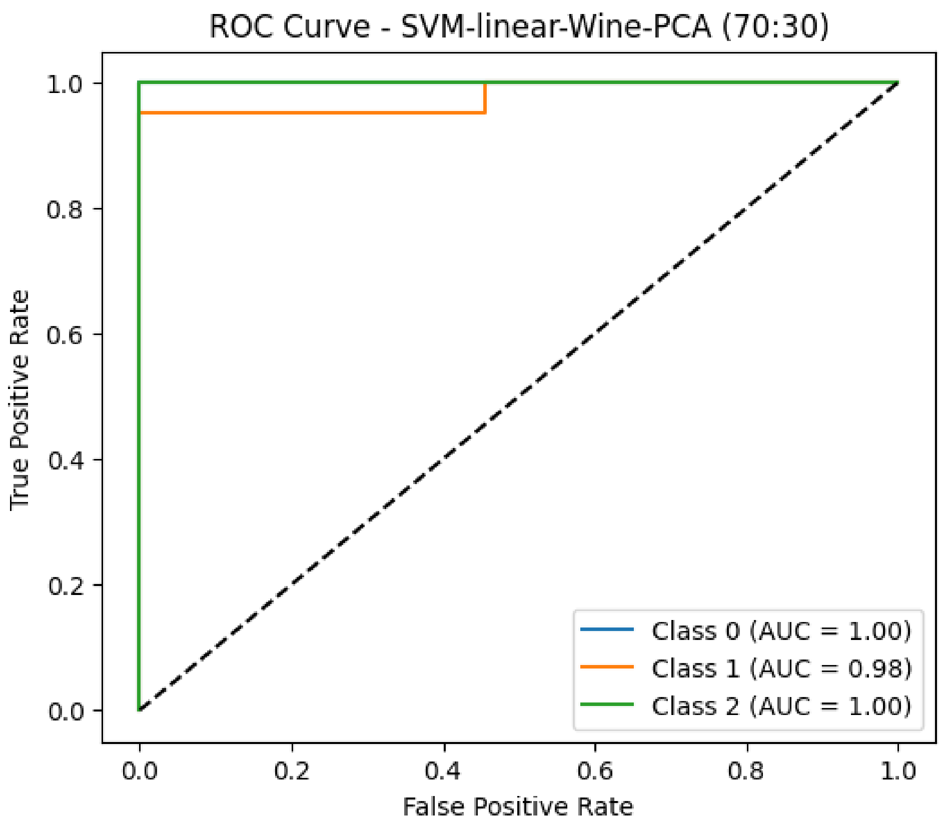


[Wine-PCA] Reduced dimensionality: 13 -> 10
SVM-linear train-test split: 70:30
[Default] SVM-linear accuracy: 0.9815
[Tuned] SVM-linear accuracy: 0.9630
[Tuned] SVM-linear best params: {'C': 0.1}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
0	SVM-linear-Wine-PCA	70:30	0.962963	0.96732	0.962963	0.963194

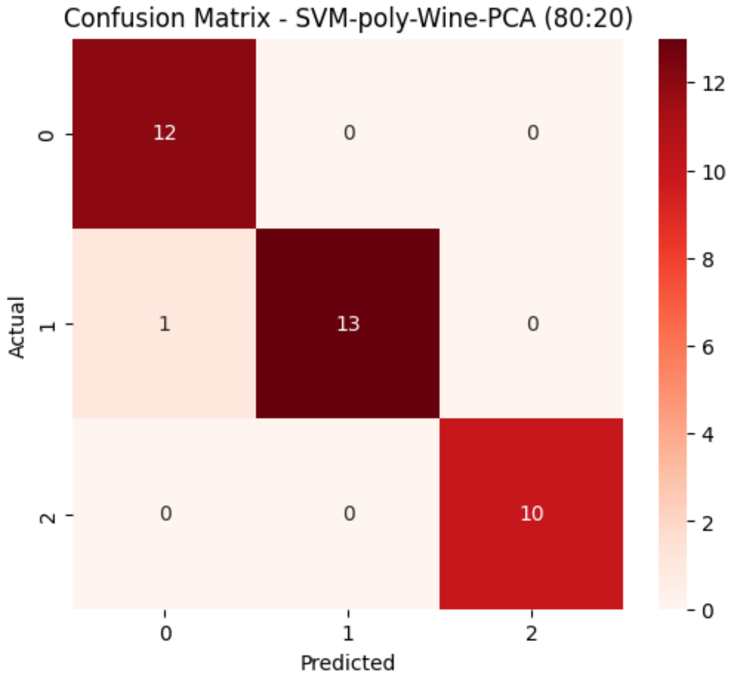


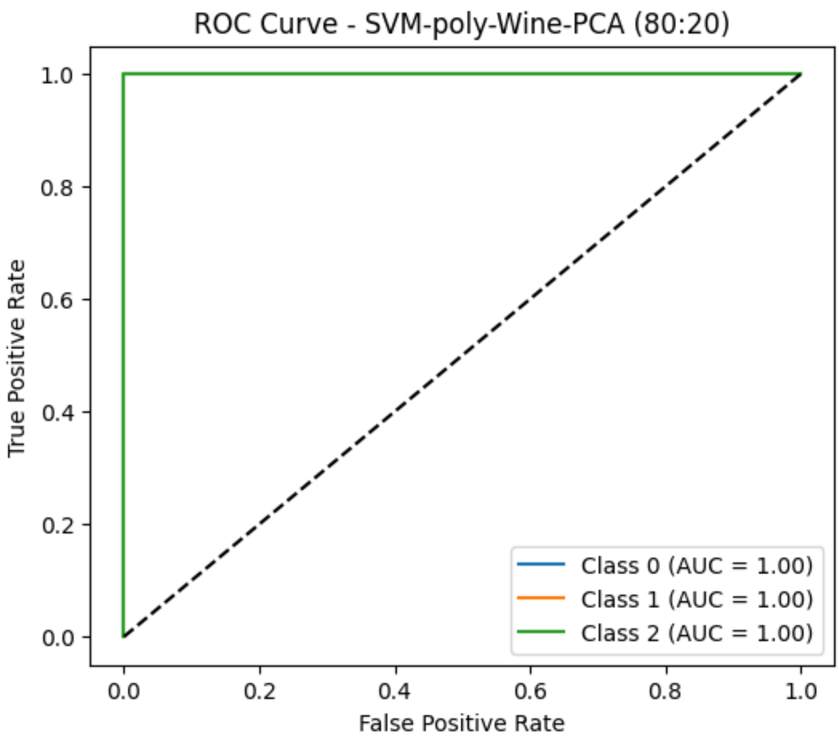


[Wine-PCA] Reduced dimensionality: 13 -> 10
SVM-poly train-test split: 80:20
[Default] SVM-poly accuracy: 1.0000
[Tuned] SVM-poly accuracy: 0.9722
[Tuned] SVM-poly best params: {'C': 10, 'degree': 3, 'gamma': 'scale'}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
0	SVM-poly-Wine-PCA	80:20	0.972222	0.974359	0.972222	0.972263

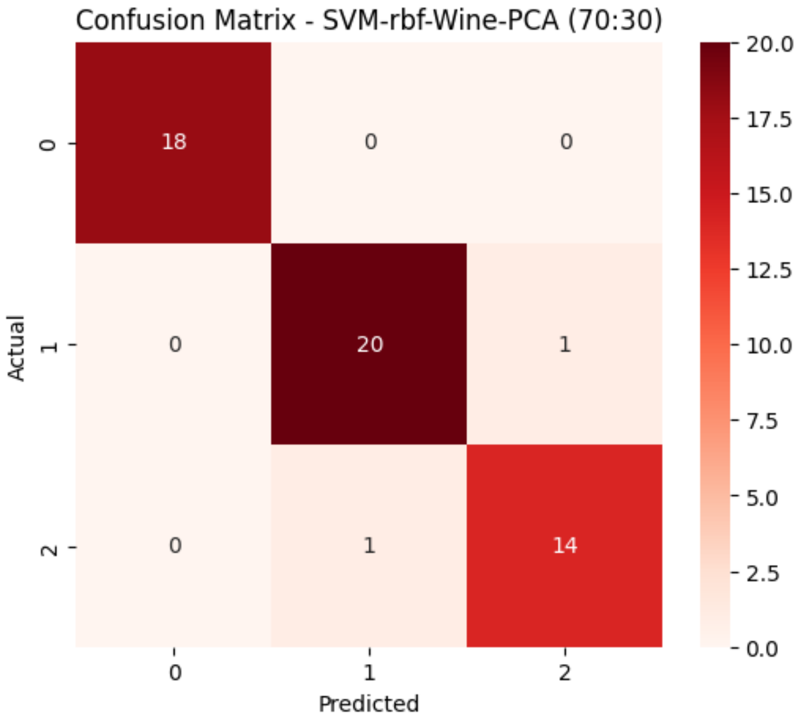


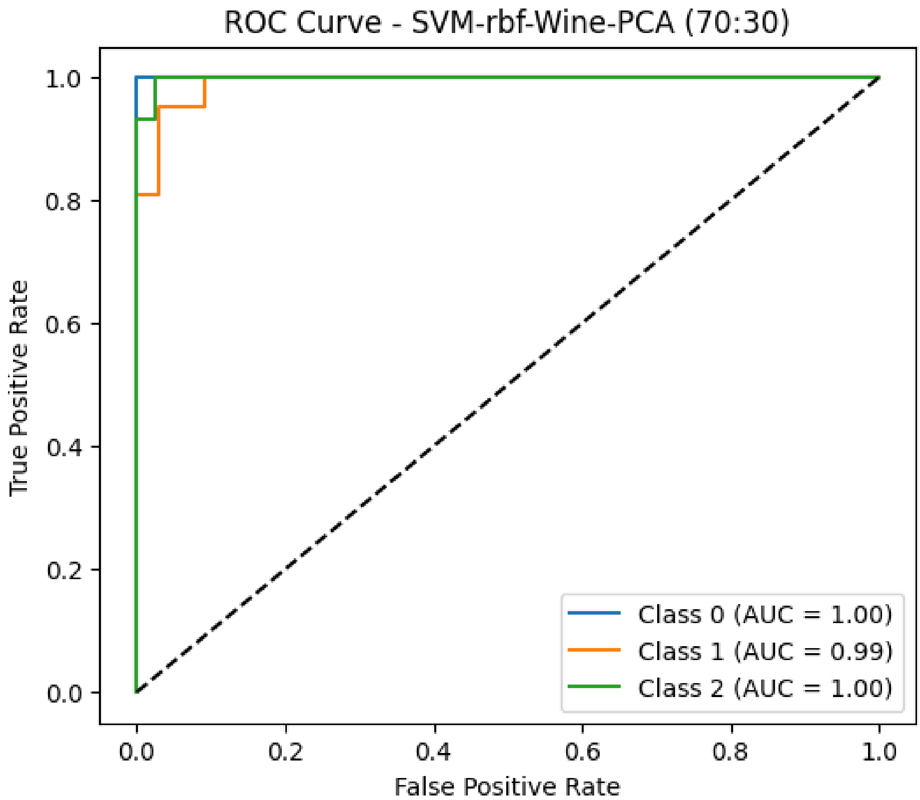


SVM-rbf train-test split: 70:30
[Default] SVM-rbf accuracy: 0.9630
[Tuned] SVM-rbf accuracy: 0.9630
[Tuned] SVM-rbf best params: {'C': 1, 'gamma': 'scale'}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
1	SVM-rbf-Wine-PCA	70:30	0.962963	0.962963	0.962963	0.962963

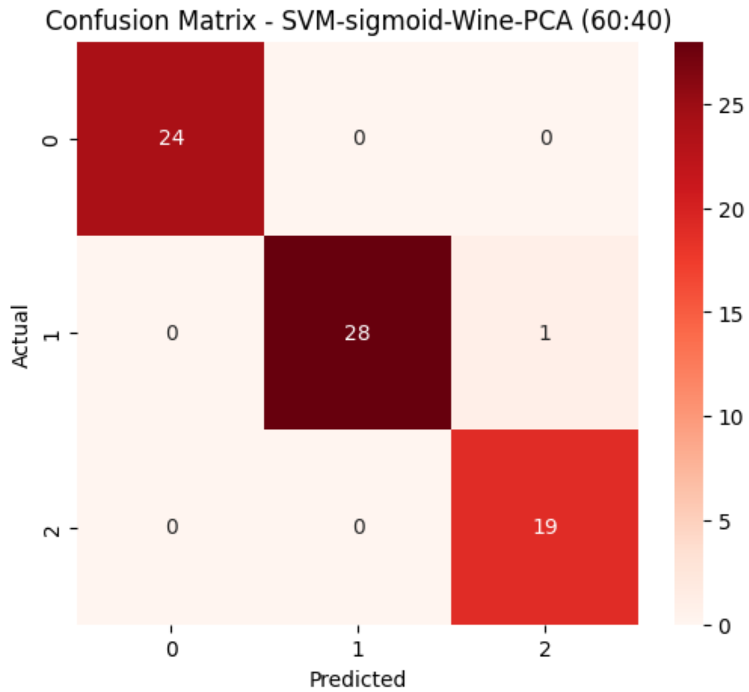


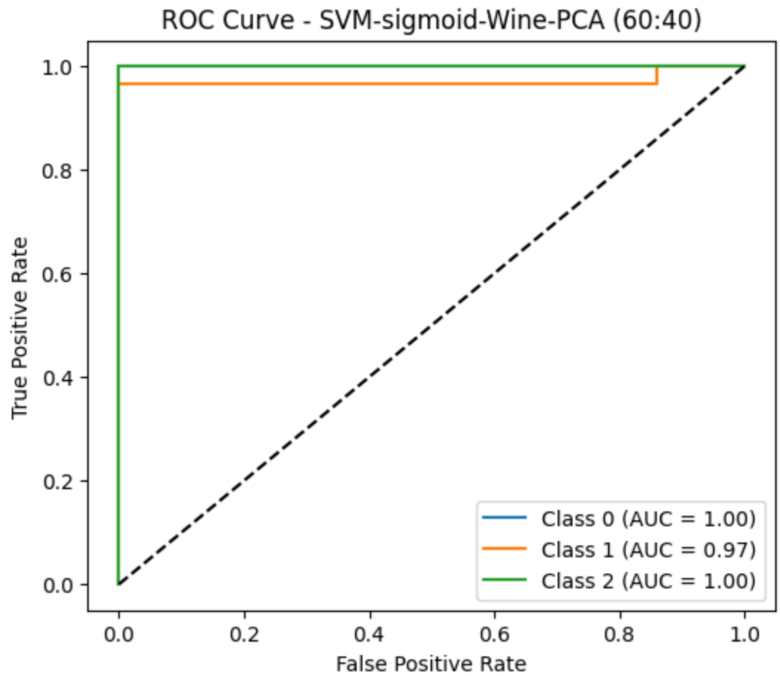


[Wine-PCA] Reduced dimensionality: 13 -> 10
SVM-sigmoid train-test split: 60:40
[Default] SVM-sigmoid accuracy: 0.9861
[Tuned] SVM-sigmoid accuracy: 0.9861
[Tuned] SVM-sigmoid best params: {'C': 1, 'gamma': 'scale'}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
0	SVM-sigmoid-Wine-PCA	60:40	0.986111	0.986806	0.986111	0.986167

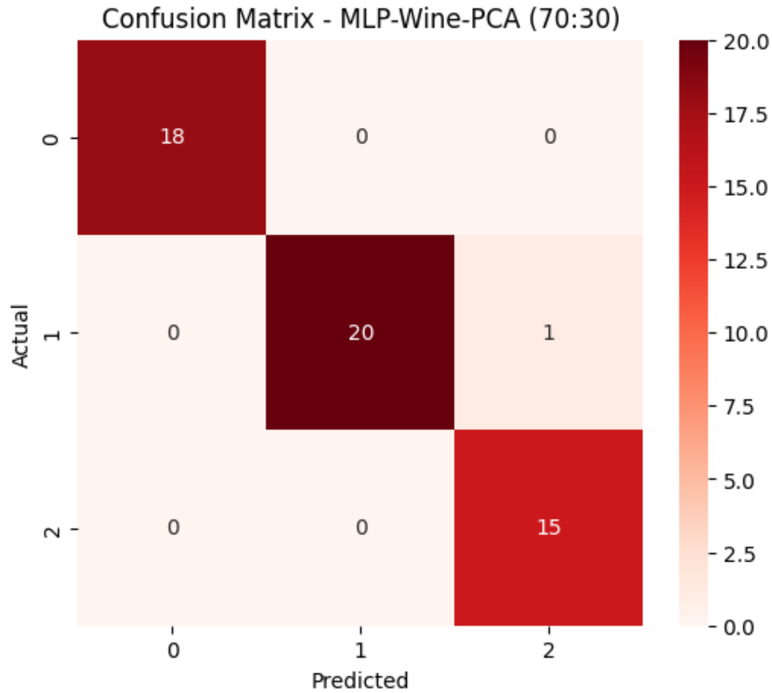


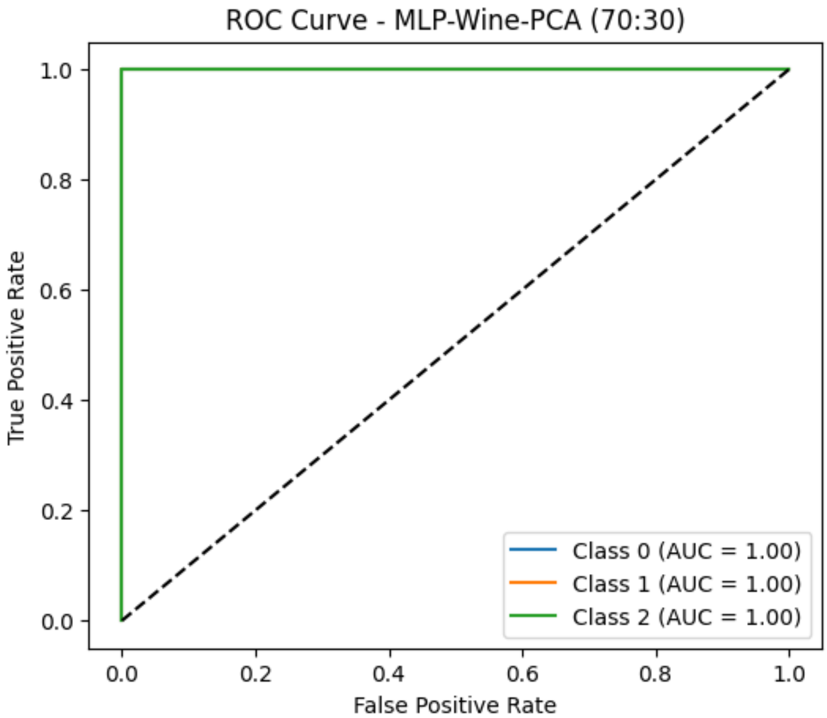


MLP train-test split: 70:30
[Default] MLP accuracy: 0.9630
[Tuned] MLP accuracy: 0.9815
[Tuned] MLP best params: {'activation': 'relu', 'hidden_layer_sizes': (100,), 'learning_rate_init': 0.01, 'max_iter': 300, 'momentum': 0.8, 'solver': 'adam'}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
2	MLP-Wine-PCA	70:30	0.981481	0.982639	0.981481	0.981554

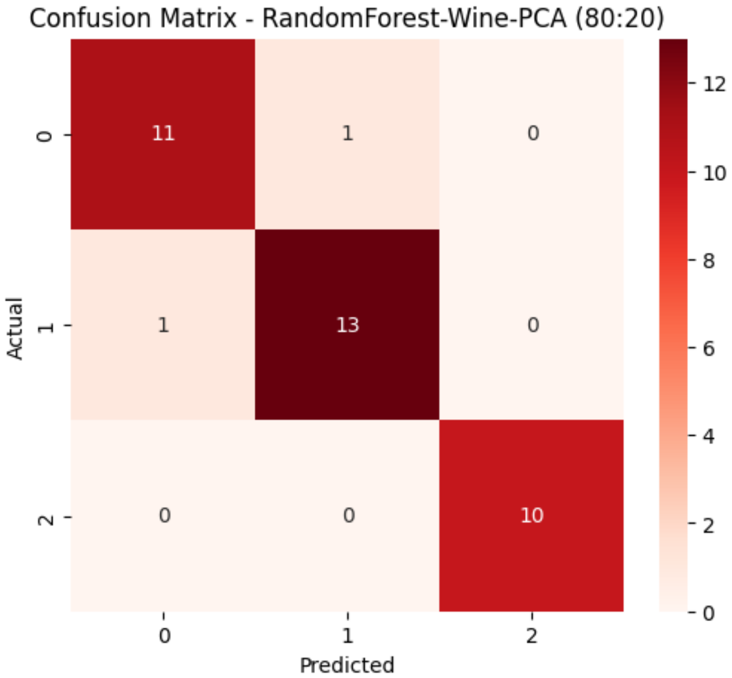


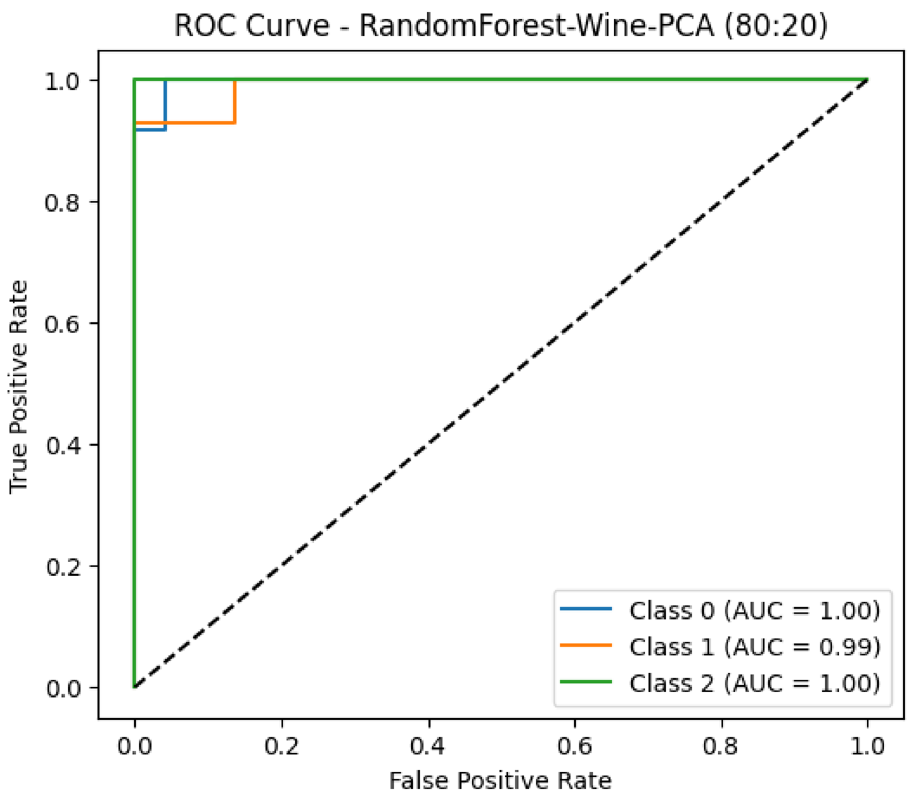


RandomForest train-test split: 80:20
[Default] RandomForest accuracy: 0.9444
[Tuned] RandomForest accuracy: 0.9444
[Tuned] RandomForest best params: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
1	RandomForest-Wine-PCA	80:20	0.944444	0.944444	0.944444	0.944444





4.2 Digits Dataset

```
digits = load_digits()
results_digits = train_and_evaluate(digits.data, digits.target, "Digits")
results_digits_pca = train_and_evaluate_with_pca(digits.data,
digits.target, "Digits")
```

Best Results:

Model	PCA	Split	Accuracy	Precision	Recall	F1-score
SVM-linear	No	80:20	0.9750	0.9754	0.9750	0.9749
SVM-poly	No	70:30	0.9759	0.9765	0.9759	0.9760
SVM-rbf	No	80:20	0.9833	0.9839	0.9833	0.9833
SVM-sigmoid	No	60:40	0.9374	0.9384	0.9374	0.9373
MLP	No	80:20	0.9806	0.9810	0.9806	0.9805
RandomForest	No	80:20	0.9639	0.9644	0.9639	0.9636
SVM-linear	Yes	80:20	0.9778	0.9779	0.9778	0.9774
SVM-poly	Yes	70:30	0.9759	0.9764	0.9759	0.9758
SVM-rbf	Yes	80:20	0.9778	0.9781	0.9778	0.9777
SVM-sigmoid	Yes	60:40	0.9319	0.9326	0.9319	0.9316
MLP	Yes	80:20	0.9750	0.9753	0.9750	0.9748
RandomForest	Yes	80:20	0.9556	0.9559	0.9556	0.9551

SVM-linear train-test split: 80:20

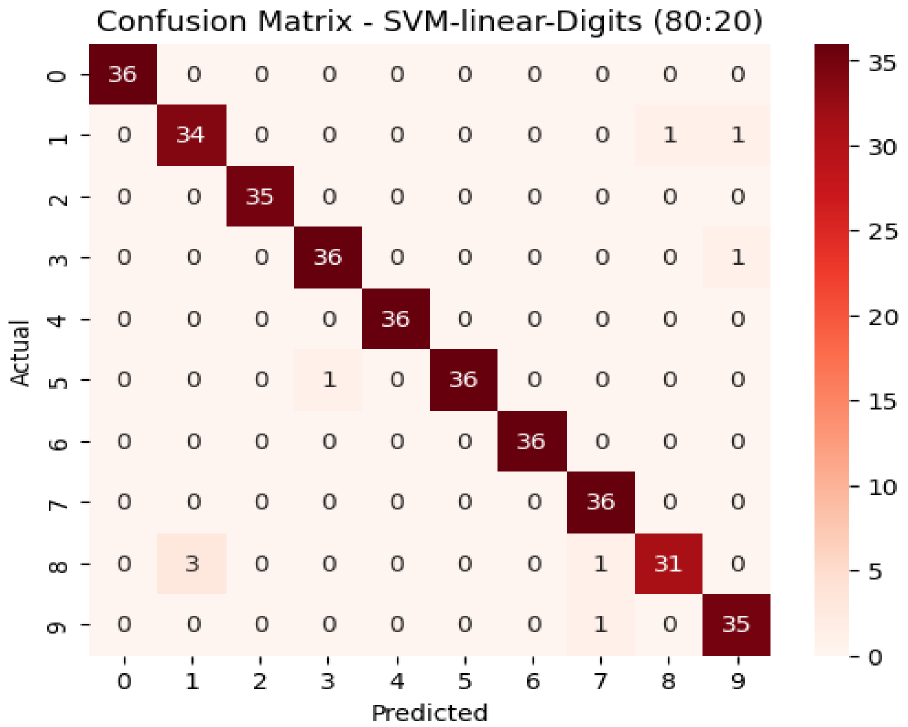
[Default] SVM-linear accuracy: 0.9750

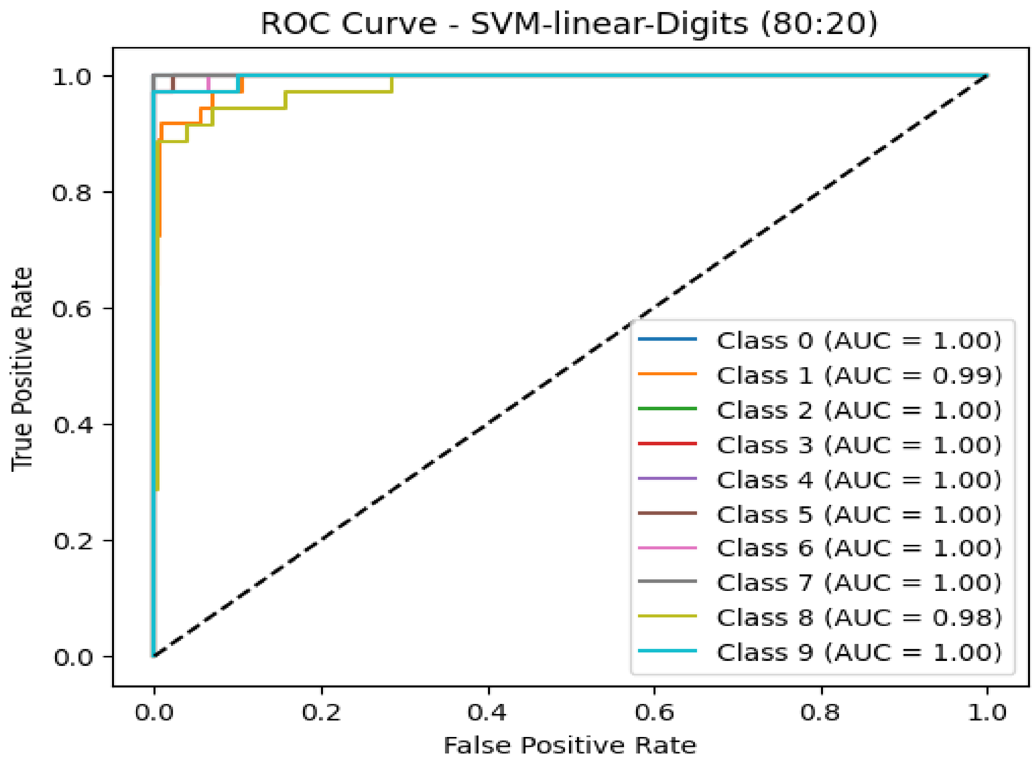
[Tuned] SVM-linear accuracy: 0.9750

[Tuned] SVM-linear best params: {'C': 1}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
0	SVM-linear-Digits	80:20	0.975	0.975407	0.975	0.974897





SVM-poly train-test split: 70:30

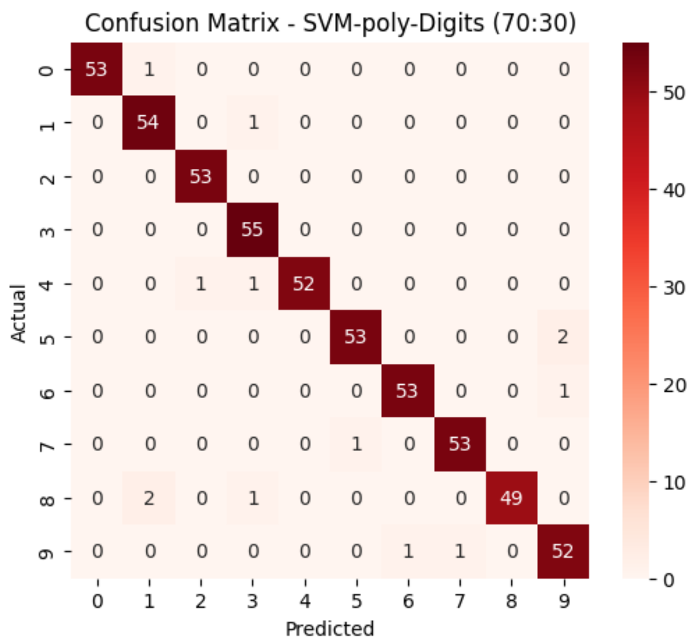
[Default] SVM-poly accuracy: 0.9667

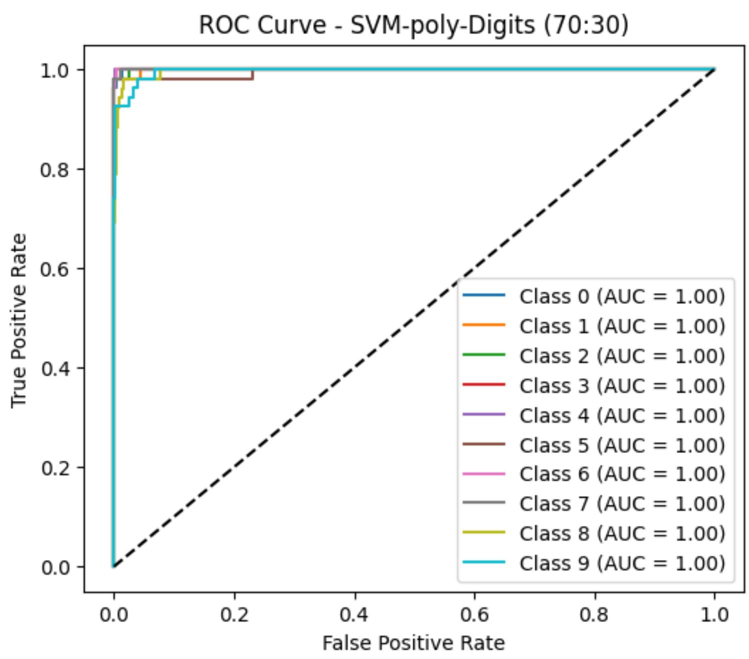
[Tuned] SVM-poly accuracy: 0.9759

[Tuned] SVM-poly best params: {'C': 10, 'degree': 2, 'gamma': 'scale'}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
0	SVM-poly-Digits	70:30	0.975926	0.976509	0.975926	0.975965





SVM-rbf train-test split: 80:20

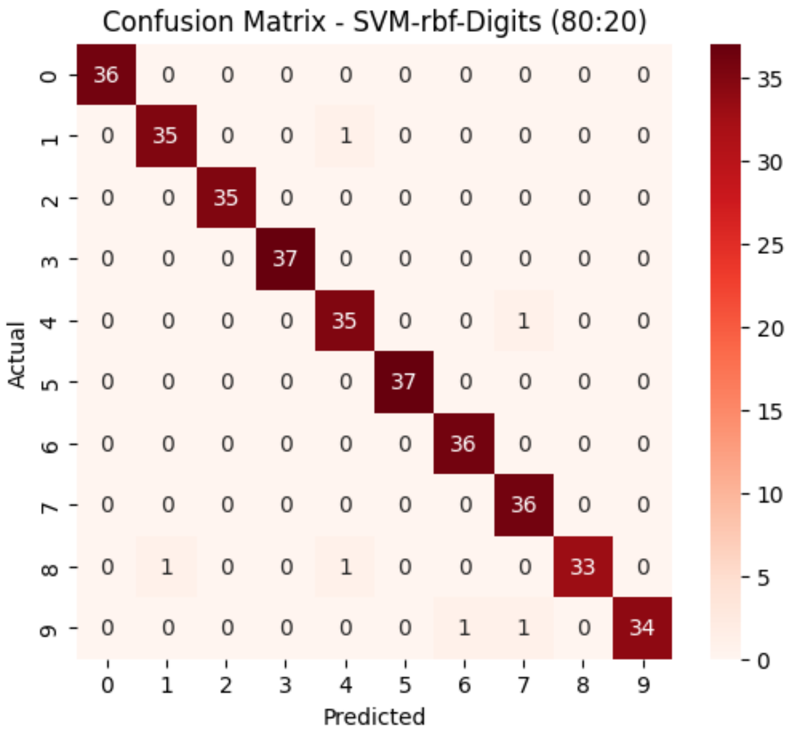
[Default] SVM-rbf accuracy: 0.9750

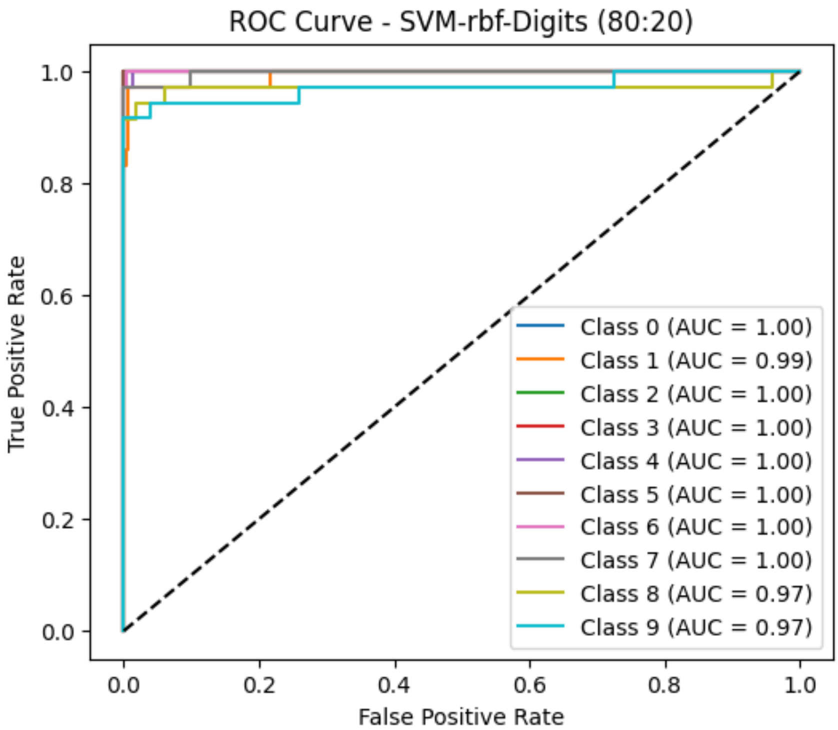
[Tuned] SVM-rbf accuracy: 0.9833

[Tuned] SVM-rbf best params: {'C': 10, 'gamma': 0.01}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
1	SVM-rbf-Digits	80:20	0.983333	0.983851	0.983333	0.983323





SVM-sigmoid train-test split: 60:40

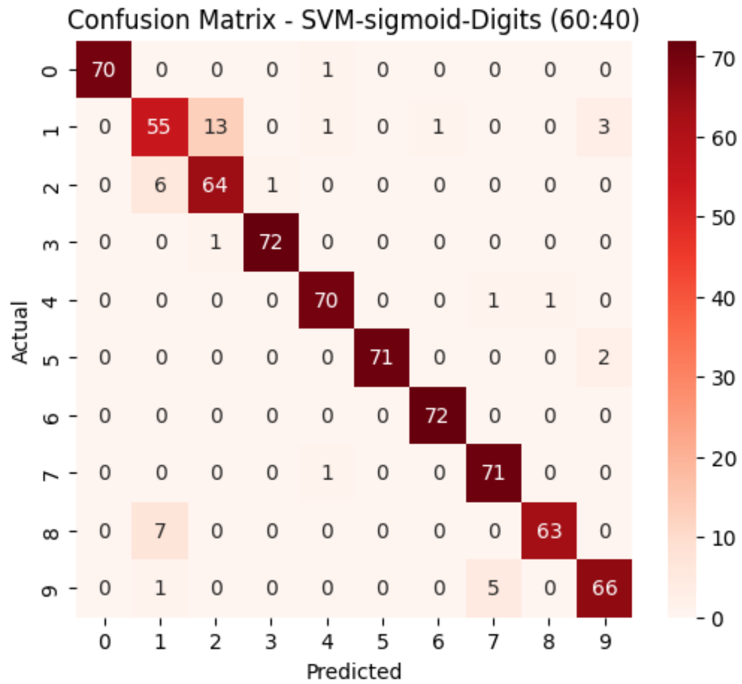
[Default] SVM-sigmoid accuracy: 0.9332

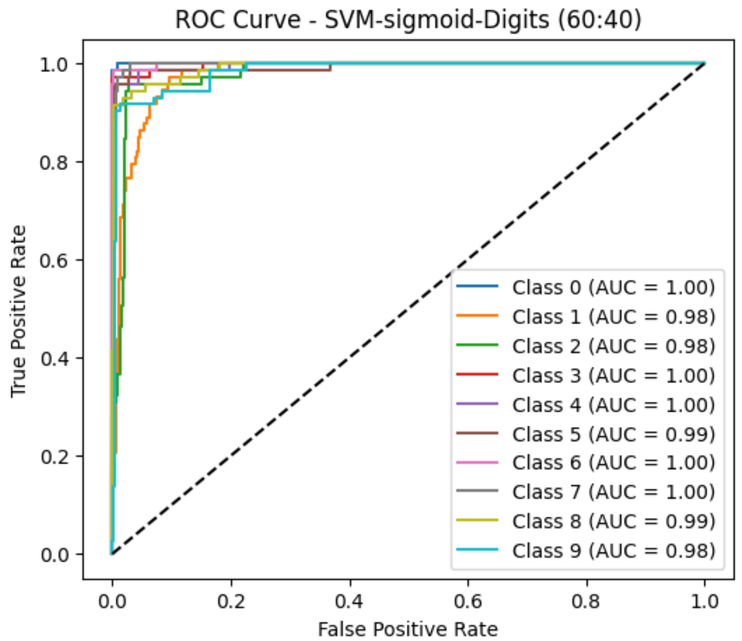
[Tuned] SVM-sigmoid accuracy: 0.9374

[Tuned] SVM-sigmoid best params: {'C': 1, 'gamma': 'auto'}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
0	SVM-sigmoid-Digits	60:40	0.937413	0.938421	0.937413	0.937338





MLP train-test split: 80:20

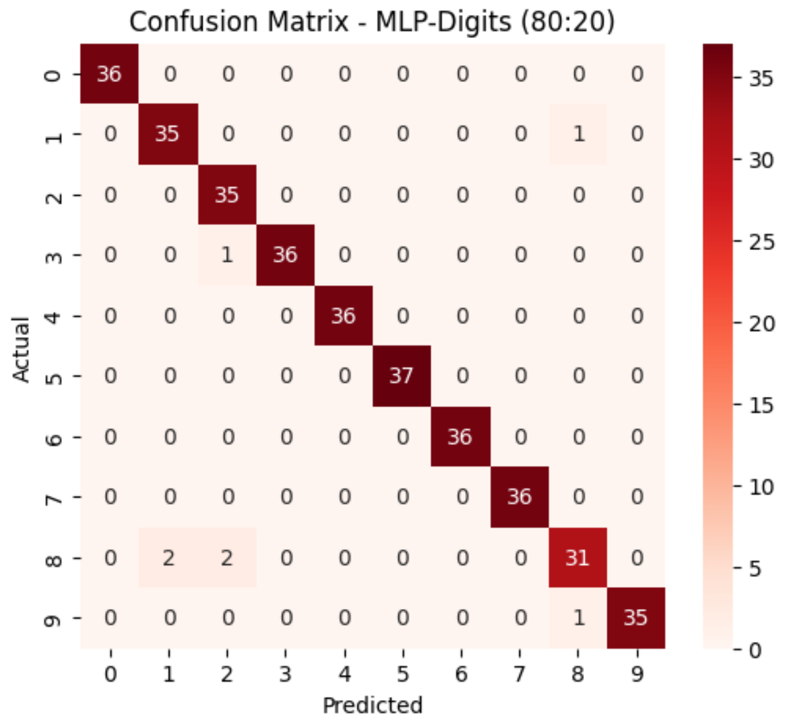
[Default] MLP accuracy: 0.9750

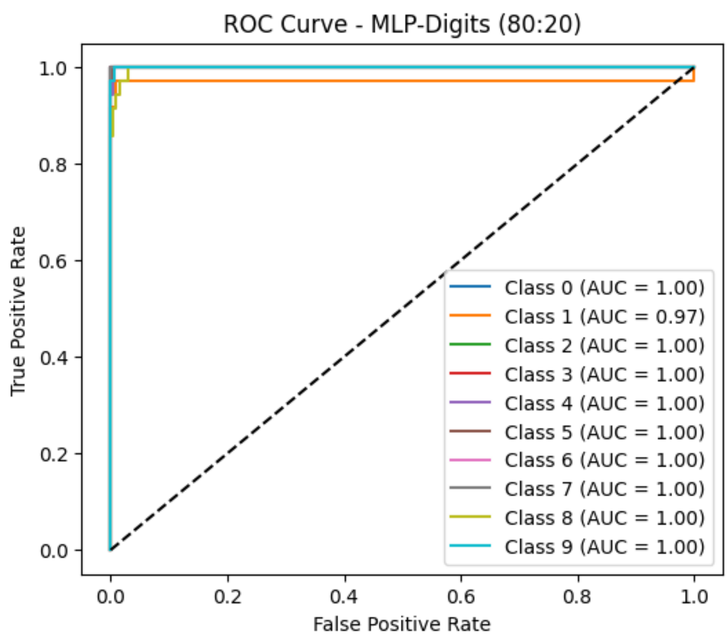
[Tuned] MLP accuracy: 0.9806

[Tuned] MLP best params: {'activation': 'tanh', 'hidden_layer_sizes': (50,), 'learning_rate_init': 0.01, 'max_iter': 300, 'momentum': 0.8, 'solver': 'adam'}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
2	MLP-Digits	80:20	0.980556	0.981027	0.980556	0.9805

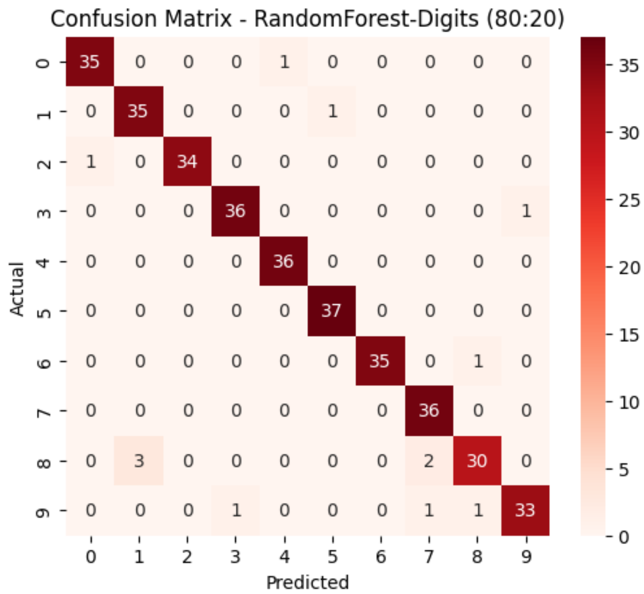


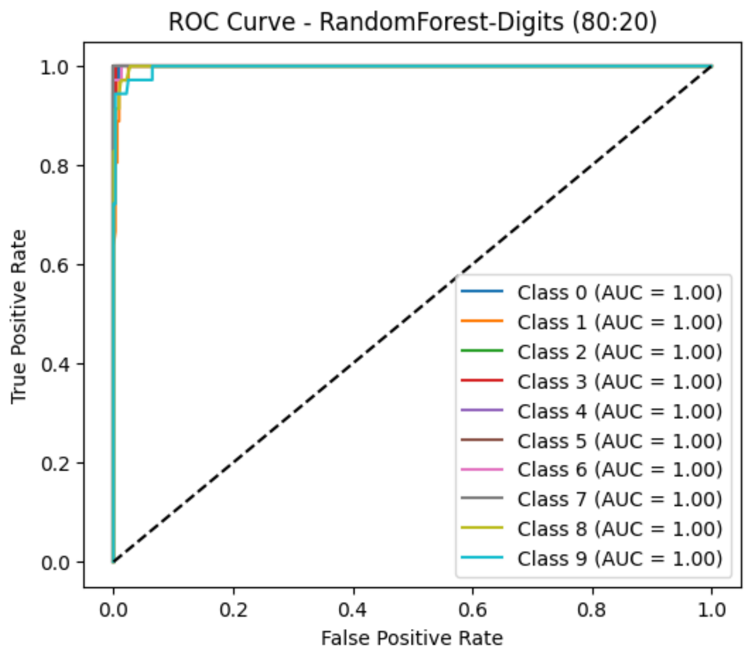


RandomForest train-test split: 80:20
[Default] RandomForest accuracy: 0.9639
[Tuned] RandomForest accuracy: 0.9639
[Tuned] RandomForest best params: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
3	RandomForest-Digits	80:20	0.963889	0.964432	0.963889	0.96361

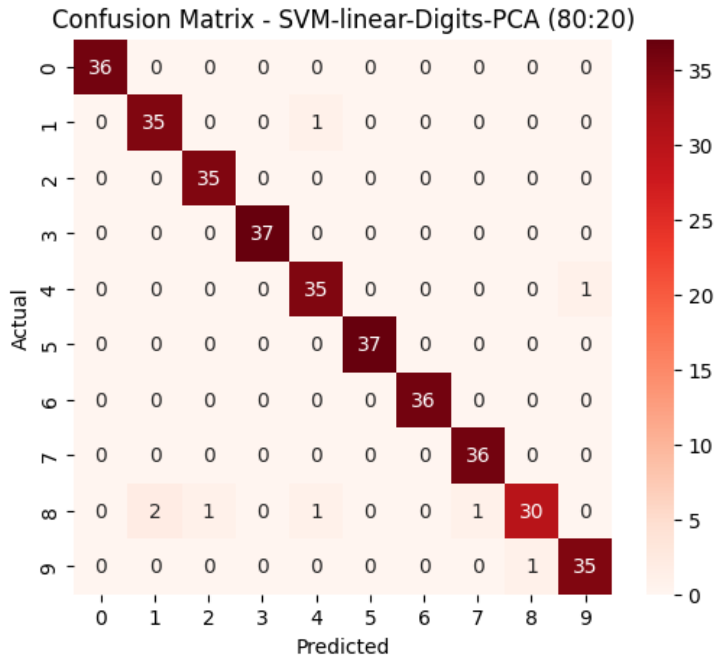


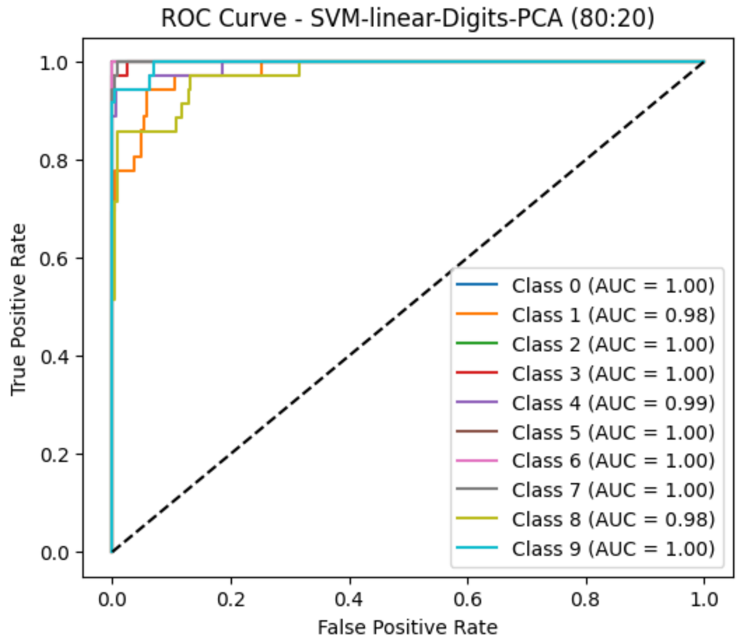


[Digits-PCA] Reduced dimensionality: 64 -> 40
SVM-linear train-test split: 80:20
[Default] SVM-linear accuracy: 0.9583
[Tuned] SVM-linear accuracy: 0.9778
[Tuned] SVM-linear best params: {'C': 0.1}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
0	SVM-linear-Digits-PCA	80:20	0.977778	0.977872	0.977778	0.977425

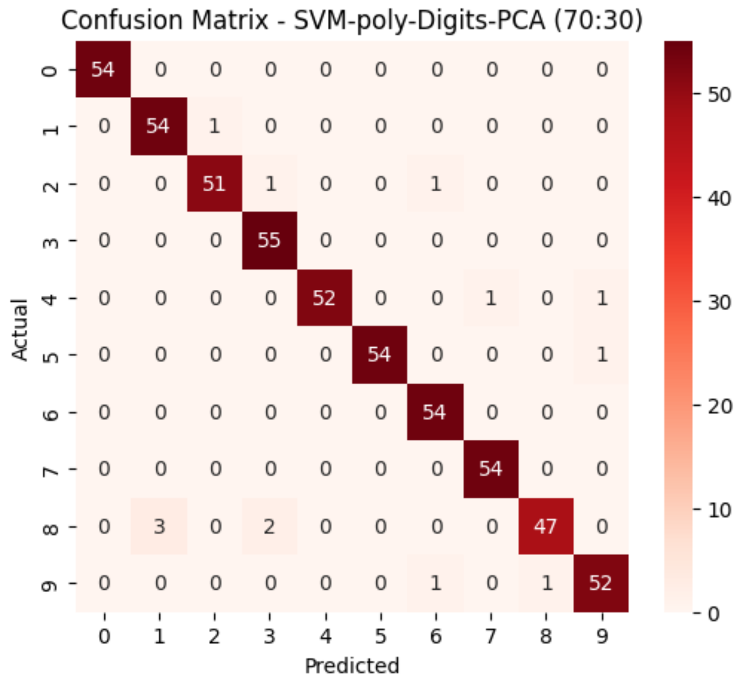


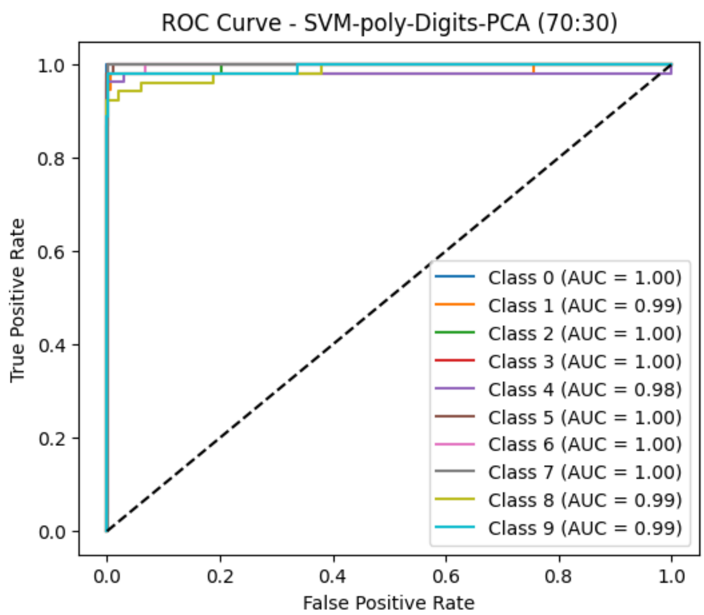


[Digits-PCA] Reduced dimensionality: 64 -> 40
SVM-poly train-test split: 70:30
[Default] SVM-poly accuracy: 0.9778
[Tuned] SVM-poly accuracy: 0.9759
[Tuned] SVM-poly best params: {'C': 10, 'degree': 3, 'gamma': 'scale'}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
0	SVM-poly-Digits-PCA	70:30	0.975926	0.976384	0.975926	0.975816





SVM-rbf train-test split: 80:20

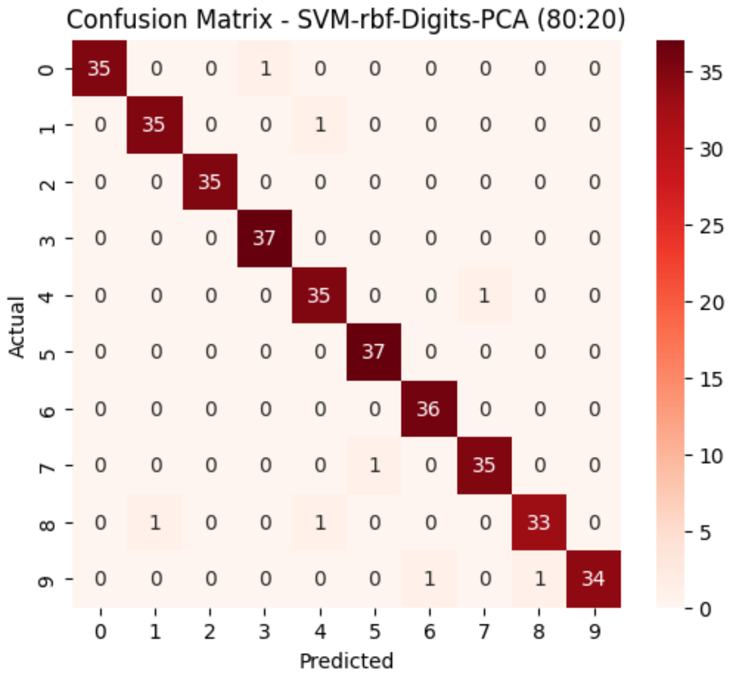
[Default] SVM-rbf accuracy: 0.9750

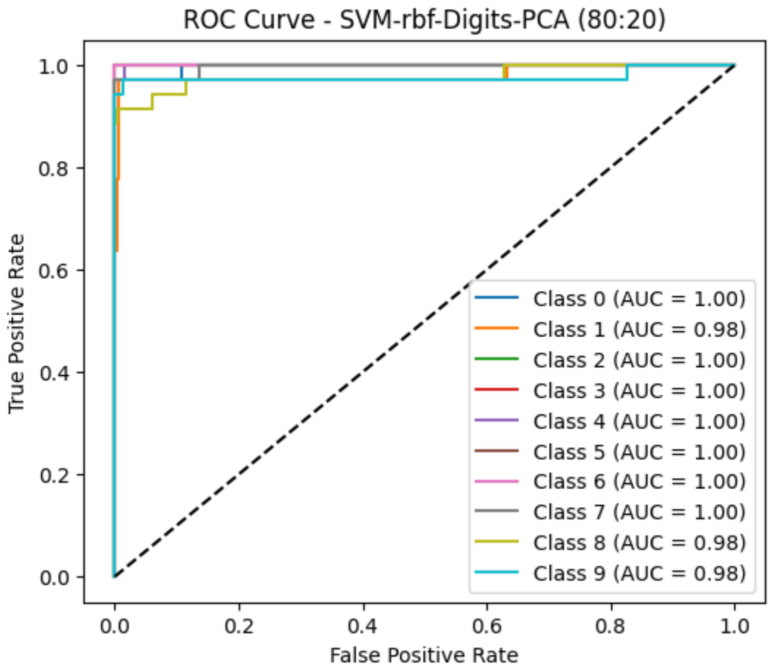
[Tuned] SVM-rbf accuracy: 0.9778

[Tuned] SVM-rbf best params: {'C': 10, 'gamma': 'scale'}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
1	SVM-rbf-Digits-PCA	80:20	0.977778	0.978068	0.977778	0.977732

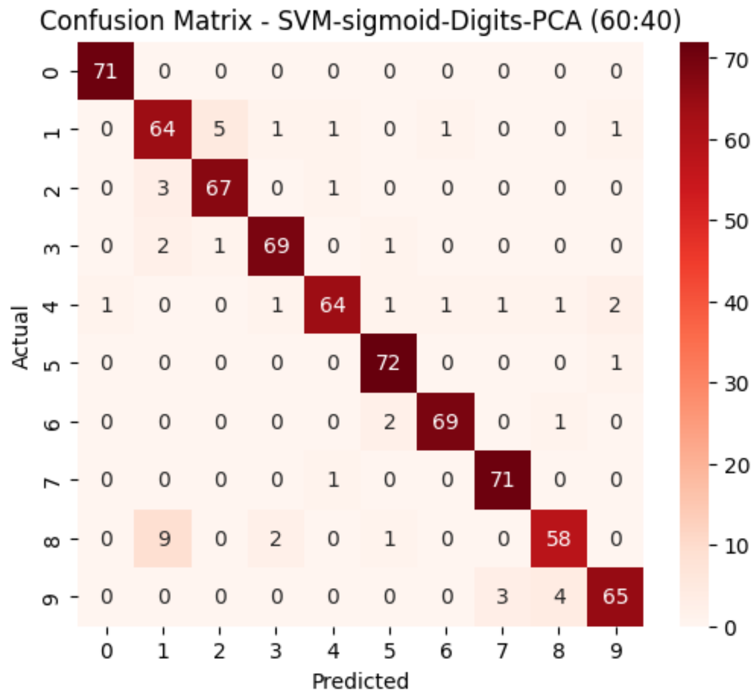


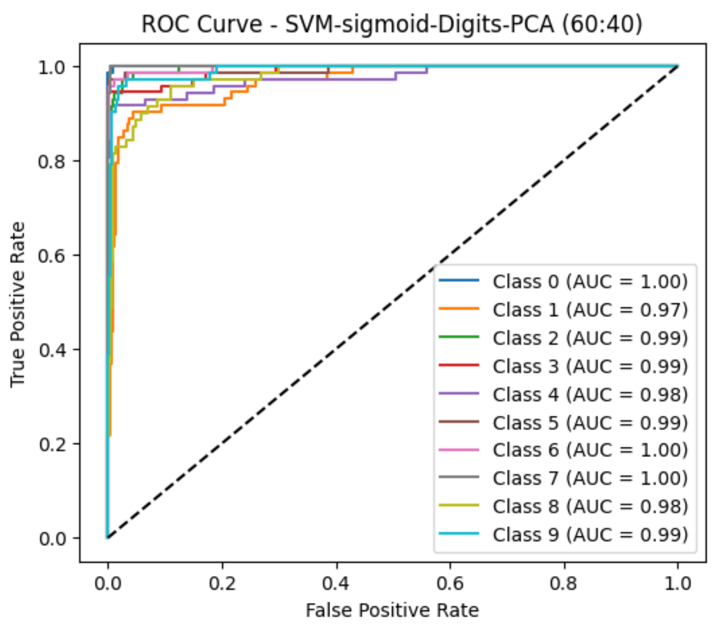


[Digits-PCA] Reduced dimensionality: 64 -> 40
SVM-sigmoid train-test split: 60:40
[Default] SVM-sigmoid accuracy: 0.9318
[Tuned] SVM-sigmoid accuracy: 0.9318
[Tuned] SVM-sigmoid best params: {'C': 1, 'gamma': 'scale'}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
0	SVM-sigmoid-Digits-PCA	60:40	0.93185	0.932555	0.93185	0.93164





MLP train-test split: 80:20

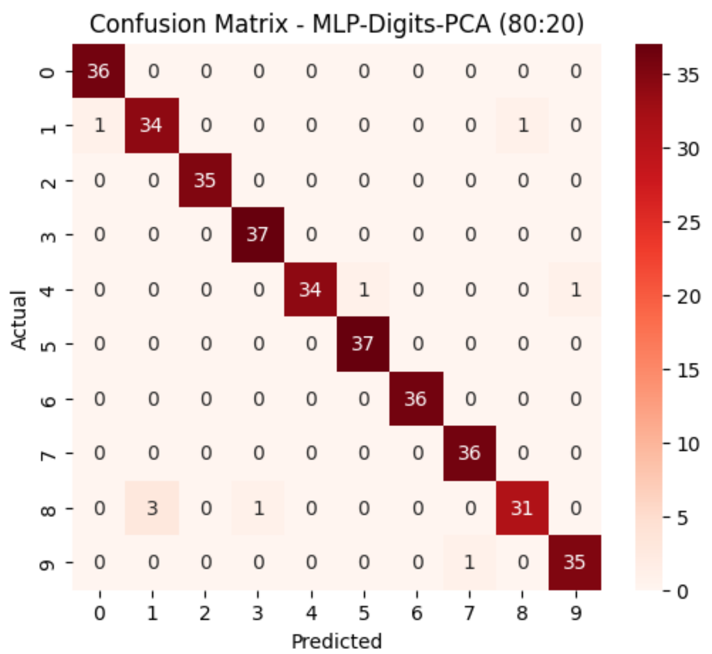
[Default] MLP accuracy: 0.9833

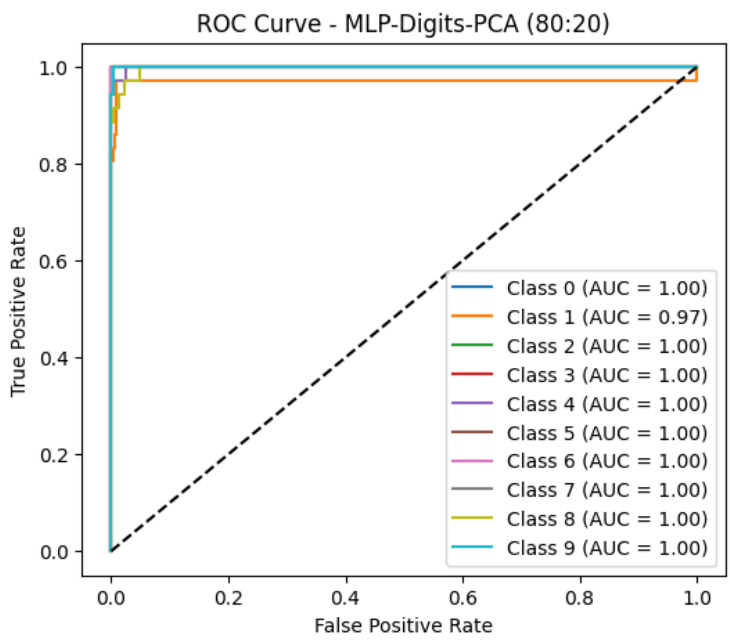
[Tuned] MLP accuracy: 0.9750

[Tuned] MLP best params: {'activation': 'relu', 'hidden_layer_sizes': (50,), 'learning_rate_init': 0.01, 'max_iter': 300, 'momentum': 0.9, 'solver': 'sgd'}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
2	MLP-Digits-PCA	80:20	0.975	0.975261	0.975	0.97478





RandomForest train-test split: 80:20

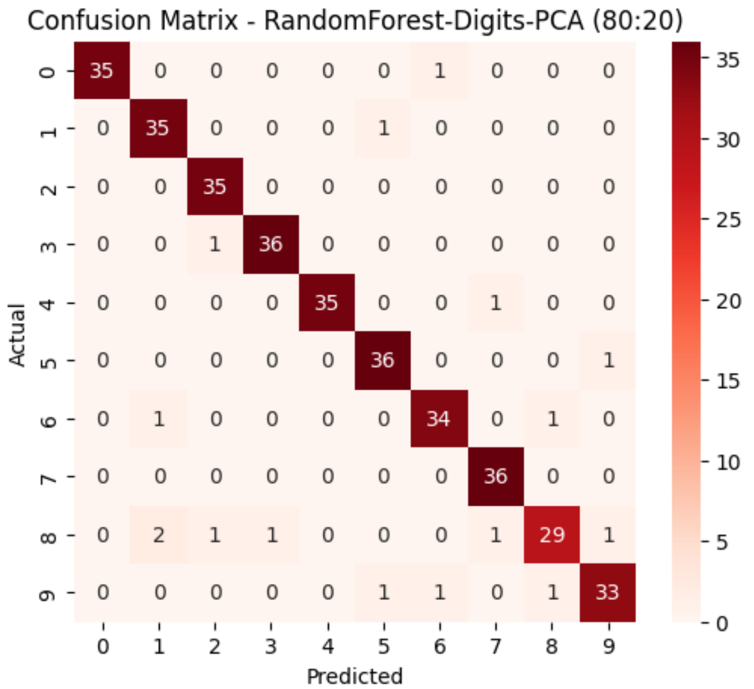
[Default] RandomForest accuracy: 0.9556

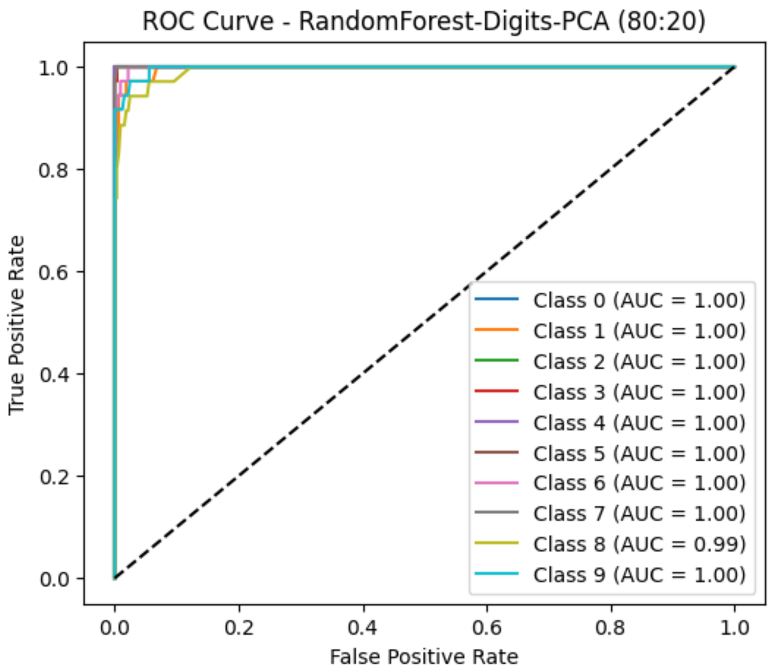
[Tuned] RandomForest accuracy: 0.9556

[Tuned] RandomForest best params: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}

Evaluation Results:

	Model	Split	Accuracy	Precision	Recall	F1-score
3	RandomForest-Digits-PCA	80:20	0.955556	0.955857	0.955556	0.955103





5. Performance Comparison

Wine Dataset (Best Results):

Classifier	Normal	PCA (0.95)
SVM-linear	0.9630	0.9630
SVM-poly	0.9167	0.9722
SVM-rbf	0.9815	0.9630
SVM-sigmoid	0.9722	0.9861
MLP	0.9815	0.9815
RandomForest	1.0000	0.9444

Digits Dataset (Best Results):

Classifier	Normal	PCA (0.95)
SVM-linear	0.9750	0.9778
SVM-poly	0.9759	0.9759
SVM-rbf	0.9833	0.9778
SVM-sigmoid	0.9374	0.9319
MLP	0.9806	0.9750
RandomForest	0.9639	0.9556

6. Conclusion

- **SVM with RBF kernel and MLP** performed best overall on the Digits dataset (≥98% accuracy).
- **Random Forest** achieved strong results on the Wine dataset.
- **PCA** slightly reduced performance but maintained competitive results while lowering feature dimensionality.
- Across splits, **70:30 and 80:20** generally produced the best outcomes.

Overall, all classifiers achieved **≥90% accuracy**.

Model	Split	Accuracy	Precision	Recall	F1-score
SVM-linear-Wine	50:50	0.9775	0.9783	0.9775	0.9774
SVM-poly-Wine	50:50	0.9213	0.9240	0.9213	0.9206
SVM-rbf-Wine	50:50	0.9775	0.9775	0.9775	0.9775
SVM-sigmoid-Wine	50:50	0.9663	0.9664	0.9663	0.9662
MLP-Wine	50:50	0.9775	0.9793	0.9775	0.9776
RandomForest-Wine	50:50	0.9775	0.9783	0.9775	0.9774
SVM-linear-Wine	60:40	0.9861	0.9867	0.9861	0.9861
SVM-poly-Wine	60:40	0.9444	0.9466	0.9444	0.9442
SVM-rbf-Wine	60:40	0.9861	0.9866	0.9861	0.9860
SVM-sigmoid-Wine	60:40	0.9722	0.9735	0.9722	0.9720
MLP-Wine	60:40	0.9861	0.9867	0.9861	0.9861
RandomForest-Wine	60:40	0.9861	0.9867	0.9861	0.9861
SVM-linear-Wine	70:30	0.9630	0.9651	0.9630	0.9626
SVM-poly-Wine	70:30	0.9444	0.9471	0.9444	0.9440
SVM-rbf-Wine	70:30	0.9815	0.9823	0.9815	0.9814
SVM-sigmoid-Wine	70:30	0.9630	0.9651	0.9630	0.9626
MLP-Wine	70:30	0.9815	0.9825	0.9815	0.9815
RandomForest-Wine	70:30	1.0000	1.0000	1.0000	1.0000
SVM-linear-Wine	80:20	0.9722	0.9741	0.9722	0.9720
SVM-poly-Wine	80:20	0.9167	0.9225	0.9167	0.9156
SVM-rbf-Wine	80:20	0.9722	0.9741	0.9722	0.9720
SVM-sigmoid-Wine	80:20	1.0000	1.0000	1.0000	1.0000

MLP-Wine	80:20	0.9444	0.9466	0.9444	0.9443
RandomForest-Wine	80:20	1.0000	1.0000	1.0000	1.0000
SVM-linear-Digits	50:50	0.9711	0.9717	0.9711	0.9710
SVM-poly-Digits	50:50	0.9800	0.9801	0.9800	0.9800
SVM-rbf-Digits	50:50	0.9711	0.9719	0.9711	0.9710
SVM-sigmoid-Digits	50:50	0.9444	0.9463	0.9444	0.9447
MLP-Digits	50:50	0.9711	0.9716	0.9711	0.9711
RandomForest-Digits	50:50	0.9577	0.9592	0.9577	0.9577
SVM-linear-Digits	60:40	0.9708	0.9721	0.9708	0.9709
SVM-poly-Digits	60:40	0.9847	0.9855	0.9847	0.9849
SVM-rbf-Digits	60:40	0.9819	0.9823	0.9819	0.9819
SVM-sigmoid-Digits	60:40	0.9374	0.9384	0.9374	0.9373
MLP-Digits	60:40	0.9736	0.9739	0.9736	0.9736
RandomForest-Digits	60:40	0.9638	0.9652	0.9638	0.9638
SVM-linear-Digits	70:30	0.9796	0.9806	0.9796	0.9795
SVM-poly-Digits	70:30	0.9759	0.9765	0.9759	0.9760
SVM-rbf-Digits	70:30	0.9815	0.9820	0.9815	0.9815
SVM-sigmoid-Digits	70:30	0.9426	0.9444	0.9426	0.9425
MLP-Digits	70:30	0.9796	0.9797	0.9796	0.9796
RandomForest-Digits	70:30	0.9648	0.9666	0.9648	0.9648
SVM-linear-Digits	80:20	0.9750	0.9754	0.9750	0.9749
SVM-poly-Digits	80:20	0.9944	0.9946	0.9944	0.9944
SVM-rbf-Digits	80:20	0.9833	0.9839	0.9833	0.9833
SVM-sigmoid-Digits	80:20	0.9472	0.9489	0.9472	0.9469

MLP-Digits	80:20	0.9806	0.9810	0.9806	0.9805
RandomForest-Digits	80:20	0.9639	0.9644	0.9639	0.9636
SVM-linear-Wine-PCA	50:50	0.9663	0.9700	0.9663	0.9665
SVM-poly-Wine-PCA	50:50	0.9551	0.9586	0.9551	0.9546
SVM-rbf-Wine-PCA	50:50	0.9888	0.9892	0.9888	0.9888
SVM-sigmoid-Wine-PCA	50:50	0.8989	0.9054	0.8989	0.8996
MLP-Wine-PCA	50:50	0.9775	0.9783	0.9775	0.9774
RandomForest-Wine-PCA	50:50	0.9101	0.9102	0.9101	0.9100
SVM-linear-Wine-PCA	60:40	0.9861	0.9868	0.9861	0.9862
SVM-poly-Wine-PCA	60:40	0.9583	0.9630	0.9583	0.9584
SVM-rbf-Wine-PCA	60:40	0.9861	0.9868	0.9861	0.9862
SVM-sigmoid-Wine-PCA	60:40	0.9861	0.9868	0.9861	0.9862
MLP-Wine-PCA	60:40	0.9722	0.9735	0.9722	0.9720
RandomForest-Wine-PCA	60:40	0.9167	0.9183	0.9167	0.9172
SVM-linear-Wine-PCA	70:30	0.9630	0.9673	0.9630	0.9632
SVM-poly-Wine-PCA	70:30	0.9630	0.9651	0.9630	0.9626
SVM-rbf-Wine-PCA	70:30	0.9630	0.9630	0.9630	0.9630
SVM-sigmoid-Wine-PCA	70:30	0.9630	0.9673	0.9630	0.9632
MLP-Wine-PCA	70:30	0.9815	0.9826	0.9815	0.9816
RandomForest-Wine-PCA	70:30	0.9074	0.9099	0.9074	0.9082
SVM-linear-Wine-PCA	80:20	0.9722	0.9741	0.9722	0.9720
SVM-poly-Wine-PCA	80:20	0.9722	0.9744	0.9722	0.9723
SVM-rbf-Wine-PCA	80:20	1.0000	1.0000	1.0000	1.0000
SVM-sigmoid-Wine-PCA	80:20	0.9722	0.9747	0.9722	0.9724

MLP-Wine-PCA	80:20	1.0000	1.0000	1.0000	1.0000
RandomForest-Wine-PCA	80:20	0.9444	0.9444	0.9444	0.9444
SVM-linear-Digits-PCA	50:50	0.9566	0.9577	0.9566	0.9564
SVM-poly-Digits-PCA	50:50	0.9600	0.9604	0.9600	0.9599
SVM-rbf-Digits-PCA	50:50	0.9778	0.9779	0.9778	0.9776
SVM-sigmoid-Digits-PCA	50:50	0.9321	0.9337	0.9321	0.9322
MLP-Digits-PCA	50:50	0.9644	0.9646	0.9644	0.9644
RandomForest-Digits-PCA	50:50	0.9344	0.9350	0.9344	0.9340
SVM-linear-Digits-PCA	60:40	0.9694	0.9697	0.9694	0.9692
SVM-poly-Digits-PCA	60:40	0.9722	0.9729	0.9722	0.9721
SVM-rbf-Digits-PCA	60:40	0.9819	0.9822	0.9819	0.9818
SVM-sigmoid-Digits-PCA	60:40	0.9318	0.9326	0.9318	0.9316
MLP-Digits-PCA	60:40	0.9722	0.9730	0.9722	0.9722
RandomForest-Digits-PCA	60:40	0.9485	0.9492	0.9485	0.9484
SVM-linear-Digits-PCA	70:30	0.9704	0.9703	0.9704	0.9701
SVM-poly-Digits-PCA	70:30	0.9759	0.9764	0.9759	0.9758
SVM-rbf-Digits-PCA	70:30	0.9759	0.9763	0.9759	0.9758
SVM-sigmoid-Digits-PCA	70:30	0.9296	0.9354	0.9296	0.9305
MLP-Digits-PCA	70:30	0.9722	0.9728	0.9722	0.9722
RandomForest-Digits-PCA	70:30	0.9537	0.9547	0.9537	0.9537
SVM-linear-Digits-PCA	80:20	0.9778	0.9779	0.9778	0.9774
SVM-poly-Digits-PCA	80:20	0.9806	0.9807	0.9806	0.9804
SVM-rbf-Digits-PCA	80:20	0.9778	0.9781	0.9778	0.9777
SVM-sigmoid-Digits-PCA	80:20	0.9306	0.9350	0.9306	0.9313

MLP-Digits-PCA	80:20	0.9750	0.9753	0.9750	0.9748
RandomForest-Digits-PCA	80:20	0.9556	0.9559	0.9556	0.9551