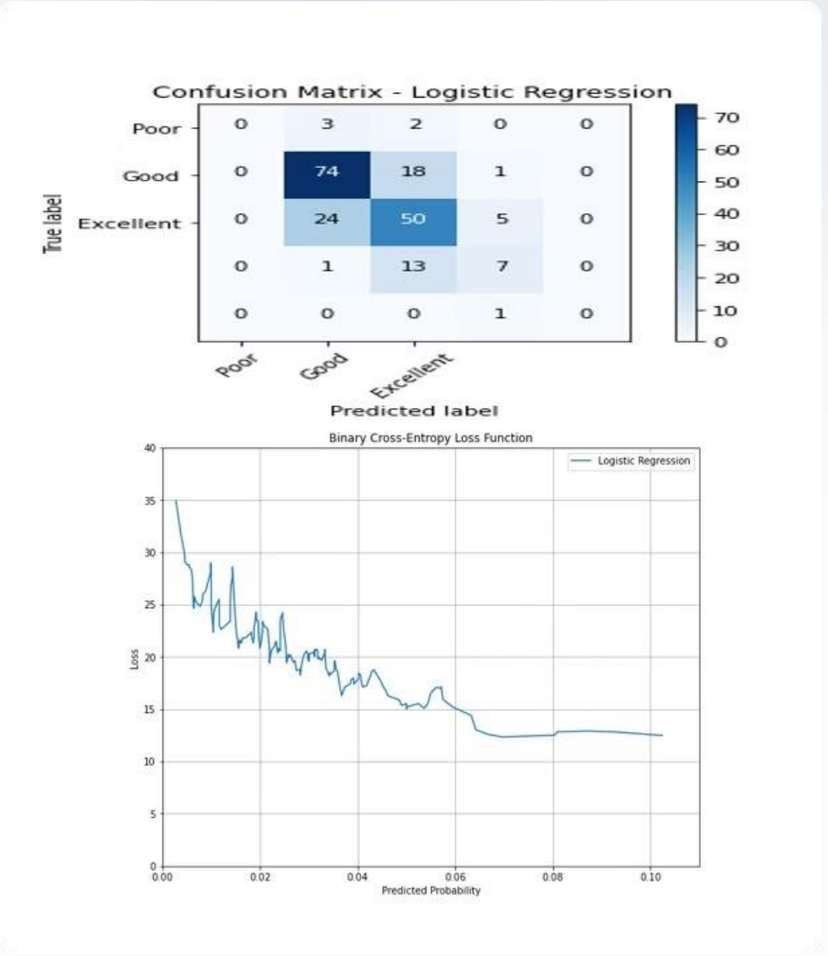
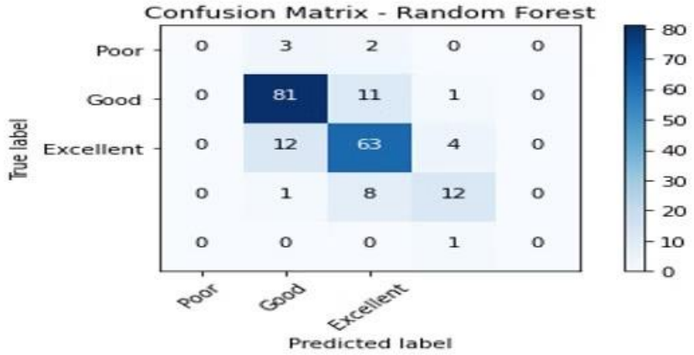
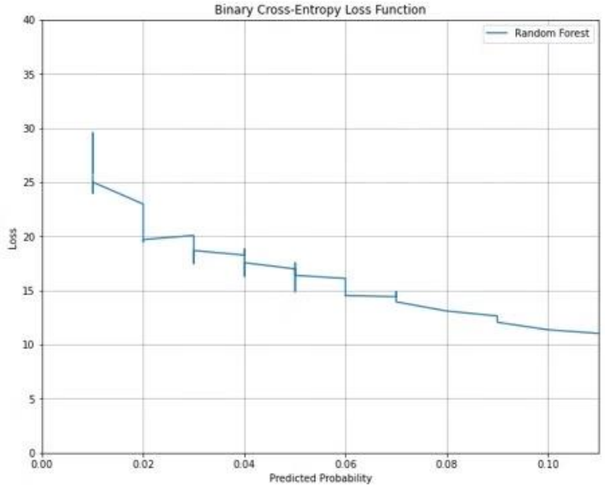
Documentation For The Pattern Pattern Recognition Project

Logistic Regression

achieved accuracy of 65.8% on the given dataset.





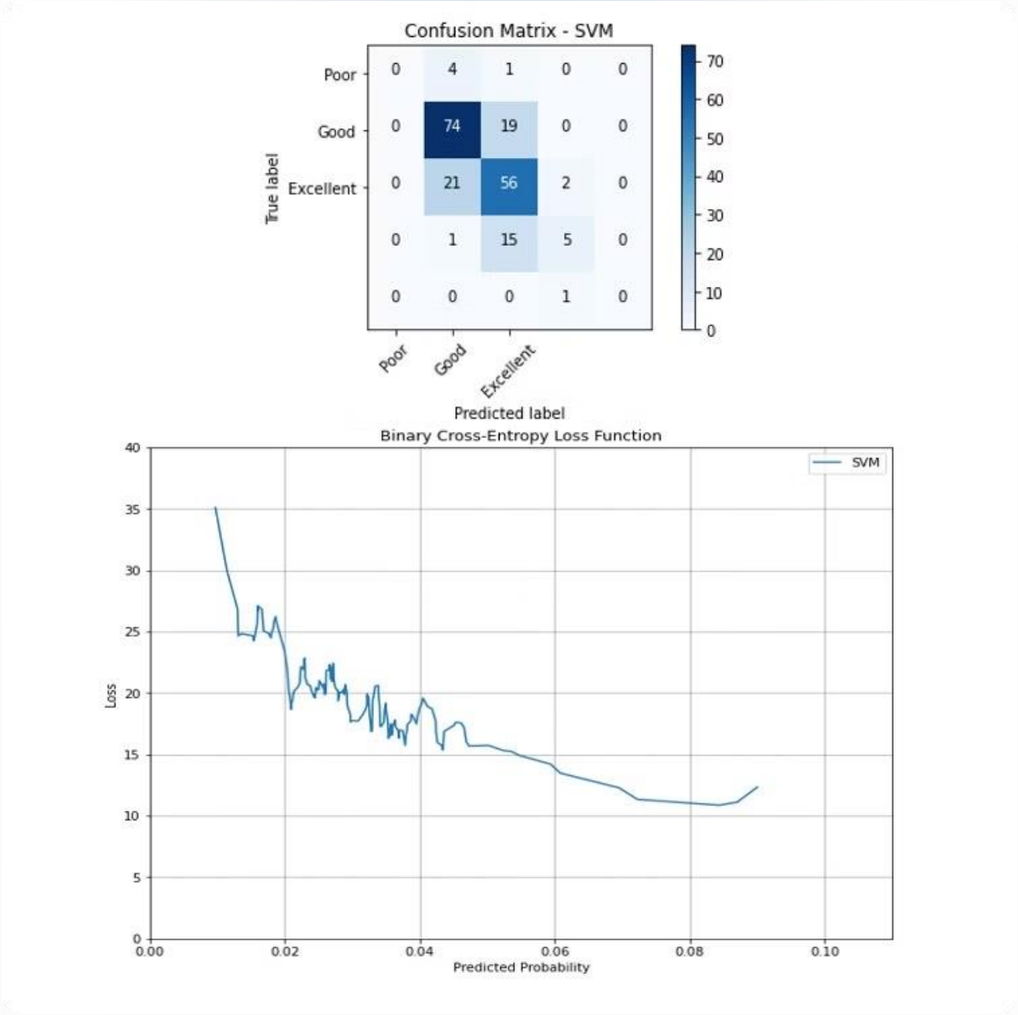


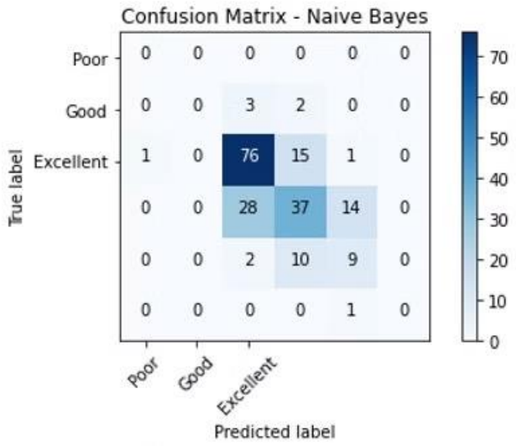
Random Forest

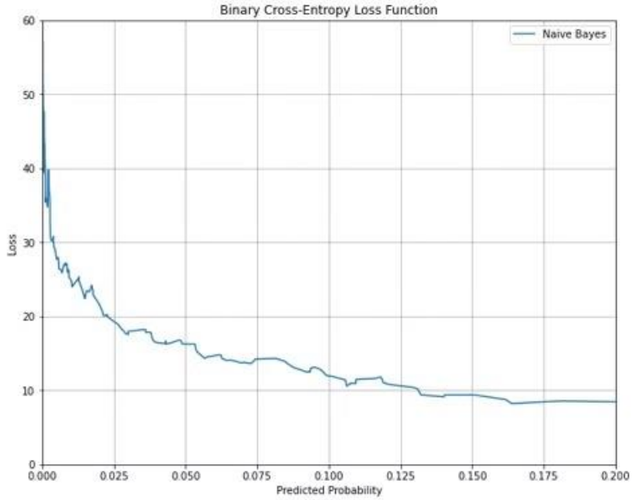
achieved accuracy of 78.9% on the given dataset.

SVM Accuracy

The SVM model achieved an accuracy of 76.4% on the given the given dataset.





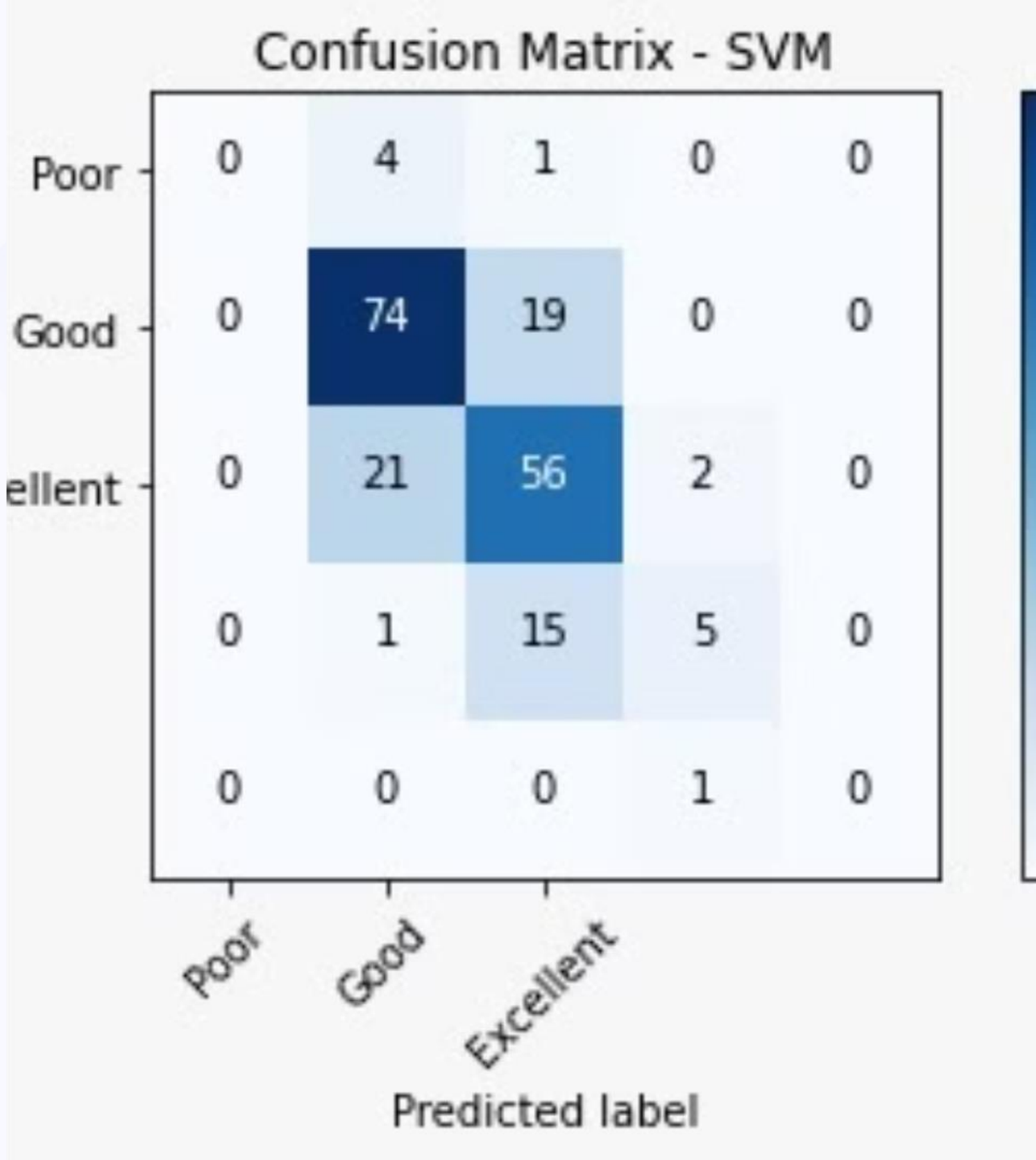


Naive Bayes Accuracy

The Naive Bayes model achieved an accuracy of 61.3% on the given dataset.

Decision Tree Accuracy

The Decision Tree model achieved an accuracy of 60.3% on the given dataset.

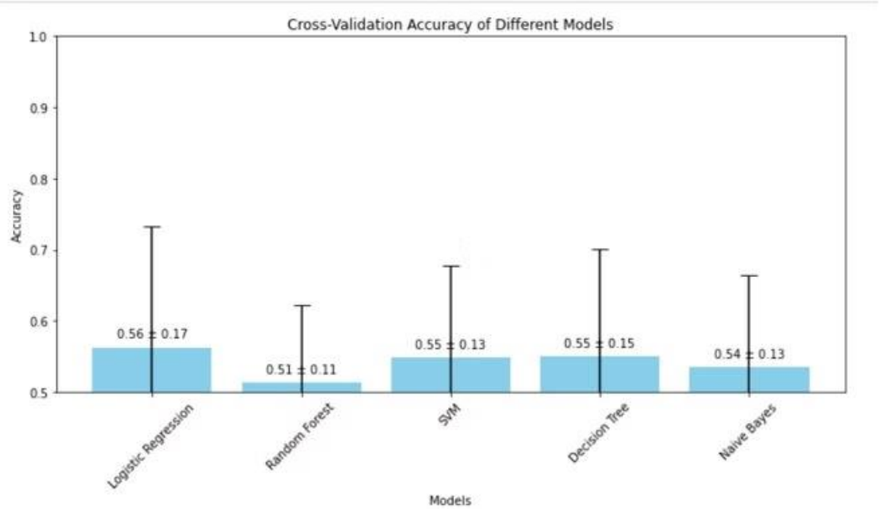


K-Neighbors Accuracy

The KNN model achieved an accuracy of 71.9% on the given dataset.

Cross Validation

Logistic Regression Cross-Validation Accuracy: 0.55 (+/- 0.10)
K neighbors Cross-Validation Accuracy: 0.55 (+/- 0.10)
Random Forest Cross-Validation Accuracy: 0.54 (+/- 0.01)
SVM Cross-Validation Accuracy: 0.48 (+/- 0.02)
Decision Tree Cross-Validation Accuracy: 0.52 (+/- 0.04)
Naive Bayes Cross-Validation Accuracy: 0.54 (+/- 0.01)



Cross-Validation Accuracy Details:
Logistic Regression: Mean Accuracy = 0.56, Standard Deviation = 0.17
Random Forest: Mean Accuracy = 0.51, Standard Deviation = 0.11
SVM: Mean Accuracy = 0.55, Standard Deviation = 0.13
Decision Tree: Mean Accuracy = 0.55, Standard Deviation = 0.15
Naive Bayes: Mean Accuracy = 0.54, Standard Deviation = 0.13

Hyperparameters Tuning

We used the grid search algorithm to find the perfect hyperparameters for our models, Unfortunately it worked well only with the SVM model but we also tinkered with the other hyperparameters and some were better left untouched.

In [22]:
```python
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC

# Define your classifier
classifier = SVC()

# Define parameters for grid search
parameters = [{'C': [1, 10, 100, 3, 5], 'kernel': ['linear']},
              {'C': [1, 30, 10, 7, 9], 'kernel': ['rbf'], 'gamma': [0.1, 0.2, 0.3, 0.4, 0.5, 0.6]}]

# Perform grid search
grid_search = GridSearchCV(estimator=classifier,
                           param_grid=parameters,
                           scoring='accuracy',
                           cv=5,
                           n_jobs=-1)

# Fit grid search to your data
grid_search.fit(X_train, y_train)

# Get the best parameters and best score
best_params = grid_search.best_params_
best_score = grid_search.best_score_
```

In [25]:
```python
accuracy = grid_search.best_score_
```

In [26]:
```python
grid_search.best_params_
```

Out[26]:
```
{'C': 1, 'gamma': 0.6, 'kernel': 'rbf'}
```

Our video

drive.google.com

Pattern Recognition - Google Dri...