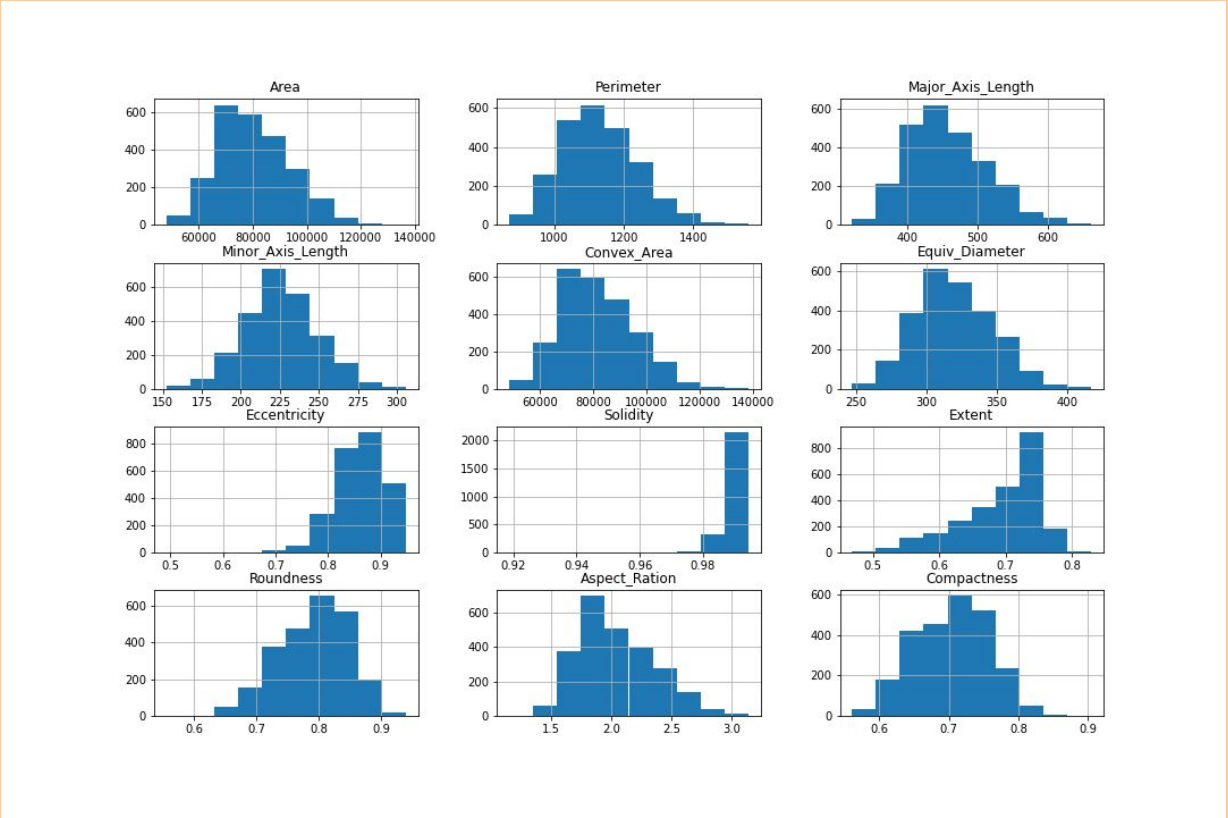
Support Vector Machine Implementation

Alistair, Joel, Nate, Peter
Group 5

Pumpkin Seeds Dataset



Model Optimization: Remove outliers

```
Testing model without outlier removal
====================================
Correctly classified Ürgüp Sivrisi: 79
Incorrectly classified Ürgüp Sivrisi: 40
Correctly classified Çerçevelik: 281
Incorrectly classified Çerçevelik: 225
              precision    recall  f1-score   support

   Çerçevelik       0.56      0.88      0.68       321
Ürgüp Sivrisi       0.66      0.26      0.37       304

     accuracy                           0.58       625
    macro avg       0.61      0.57      0.53       625
 weighted avg       0.61      0.58      0.53       625

 Matthews Coefficient: 0.17218803968747992
```

```
Testing model with outlier removal
====================================
Correctly classified Ürgüp Sivrisi: 65
Incorrectly classified Ürgüp Sivrisi: 31
Correctly classified Çerçevelik: 296
Incorrectly classified Çerçevelik: 213
              precision    recall  f1-score   support

   Çerçevelik       0.58      0.91      0.71       327
Ürgüp Sivrisi       0.68      0.23      0.35       278

     accuracy                           0.60       605
    macro avg       0.63      0.57      0.53       605
 weighted avg       0.63      0.60      0.54       605

 Matthews Coefficient: 0.18960649213208658
```

Model Optimization: Scaling Data

```
Testing model without scaling
============================
Correctly classified Ürgüp Sivrisi: 79
Incorrectly classified Ürgüp Sivrisi: 40
Correctly classified Çerçevelik: 281
Incorrectly classified Çerçevelik: 225
              precision    recall  f1-score   support

   Çerçevelik       0.56      0.88      0.68       321
Ürgüp Sivrisi       0.66      0.26      0.37       304

     accuracy                           0.58       625
    macro avg       0.61      0.57      0.53       625
 weighted avg       0.61      0.58      0.53       625

 Matthews Coefficient: 0.17218803968747992
```

```
Testing model with scaling
==========================
Correctly classified Ürgüp Sivrisi: 269
Incorrectly classified Ürgüp Sivrisi: 20
Correctly classified Çerçevelik: 284
Incorrectly classified Çerçevelik: 52
              precision    recall  f1-score   support

   Çerçevelik       0.85      0.93      0.89       304
Ürgüp Sivrisi       0.93      0.84      0.88       321

     accuracy                           0.88       625
    macro avg       0.89      0.89      0.88       625
 weighted avg       0.89      0.88      0.88       625

 Matthews Coefficient: 0.7741229971114867
```

Model Optimization: Removing Features with Low Correlation

```
Testing model with all features inlcuded
=========================================
Correctly classified Ürgüp Sivrisi: 79
Incorrectly classified Ürgüp Sivrisi: 40
Correctly classified Çerçevelik: 281
Incorrectly classified Çerçevelik: 225
              precision    recall  f1-score   support

   Çerçevelik       0.56      0.88      0.68       321
Ürgüp Sivrisi       0.66      0.26      0.37       304

     accuracy                           0.58       625
    macro avg       0.61      0.57      0.53       625
 weighted avg       0.61      0.58      0.53       625

 Matthews Coefficient: 0.17218803968747992
```

```
Testing model after removing low correlation features
=====================================================
Correctly classified Ürgüp Sivrisi: 200
Incorrectly classified Ürgüp Sivrisi: 49
Correctly classified Çerçevelik: 272
Incorrectly classified Çerçevelik: 104
              precision    recall  f1-score   support

   Çerçevelik       0.72      0.85      0.78       321
Ürgüp Sivrisi       0.80      0.66      0.72       304

     accuracy                           0.76       625
    macro avg       0.76      0.75      0.75       625
 weighted avg       0.76      0.76      0.75       625

 Matthews Coefficient: 0.5158212751160379
```

Regularization (C)

- Optimal C value was when C = 9.0
- Value was increased from the default (1.0) so the data points would be classified more accurately
    - Too much of an increase can cause overfitting
- Cleaning the data allows the C value to remain relatively small while still producing accurate results

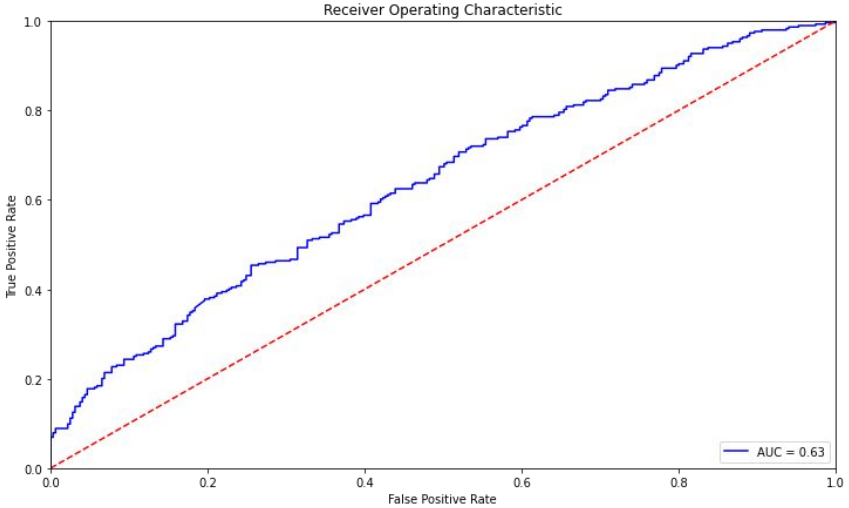
Kernel & Gamma

- Combining the optimal C value of 9.0, various kernels were explored to see which one gave the most optimal model (linear, rbf, and polynomial)
- Gamma was also looked at (default value=”scale”)
  - The most optimum range that resulted in higher scores are 1.0-1.1
  - Gamma value is important in helping to define how strong of a fit is used in kernels
  - However, is only applicable for RBF and Polynomial
- The following results are what were the most optimum values for each kernel:
  - RBF (gamma = 1.0, C=9.0)
  - Linear()
  - Polynomial(degree=3, gamma=1.1, C=9.0)

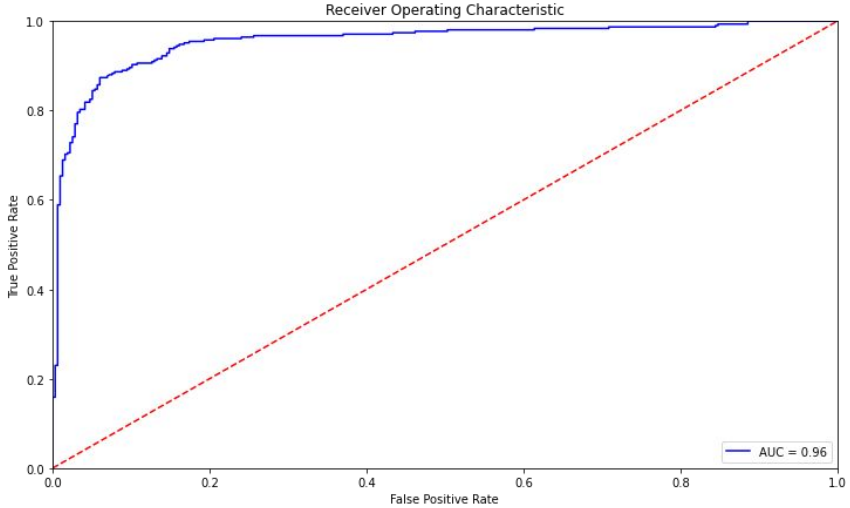
ROC AUC

Default parameters:

- kernel = ‘rbf’
- C = 1
- gamma = ‘scale’



Before standardizing

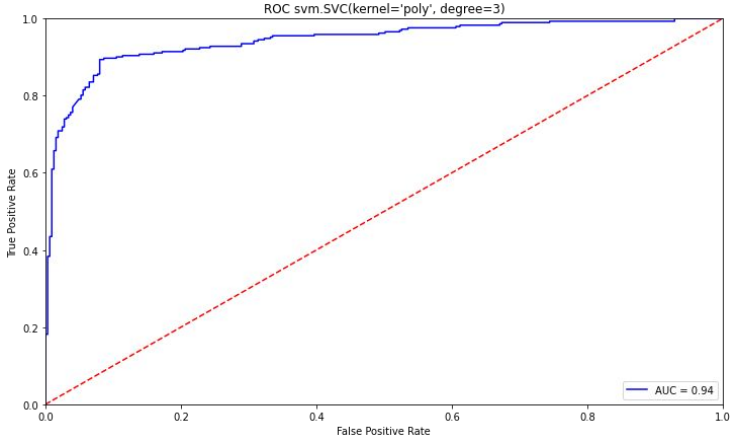


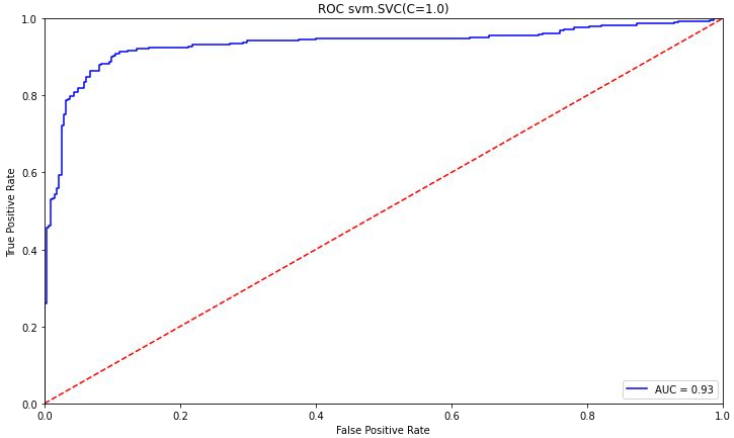
After standardizing

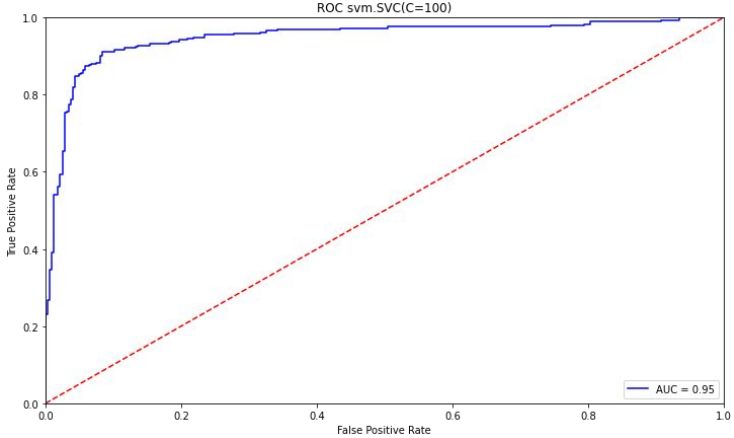
cleaning and scaling

```
data = pd.read_excel("https://github.com/plmorris/XGBoost-Group-5/blob/main/Implementation/Pumpkin_Seeds_Dataset.xlsx?raw=true")

# remove low correlation features
data_high_corr = data.loc[:,['Major_Axis_Length', 'Eccentricity', 'Roundness', 'Aspect_Ration', 'Compactness', 'Class']]

# remove outliers for each column if value is greater than 3 standard deviations from the mean
# change each outlier to np.nan
for column in data_high_corr.columns:
    if (pd.api.types.is_numeric_dtype(data_high_corr[column])):
        mean = data_high_corr[column].mean()
        std = data_high_corr[column].std()
        data_high_corr[column] = data_high_corr[column].apply(lambda x: np.nan if (x < (mean - 3 * std)) | (x > (mean + 3 * std)) else x)
# create new df with outlier rows cleaned
data_clean = data_high_corr.dropna()

data_clean = pd.get_dummies(data_clean,columns=['Class'],drop_first=True)

# separate X and y
y = data_clean["Class_Ürgüp Sivrisi"]
X = data_clean.drop(["Class_Ürgüp Sivrisi"], axis=1)
# train-test split with scaling
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25)
scaler=StandardScaler()
scaler.fit(X_train)
X_train=scaler.transform(X_train)
X_test=scaler.transform(X_test)
```

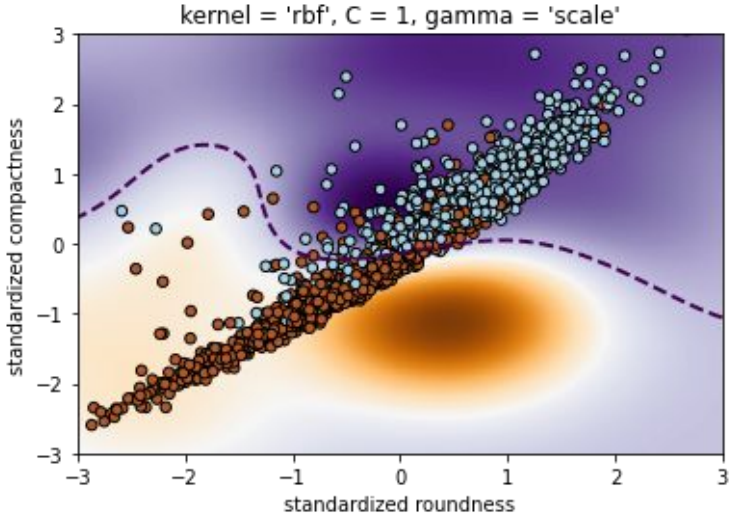
ROC AUC

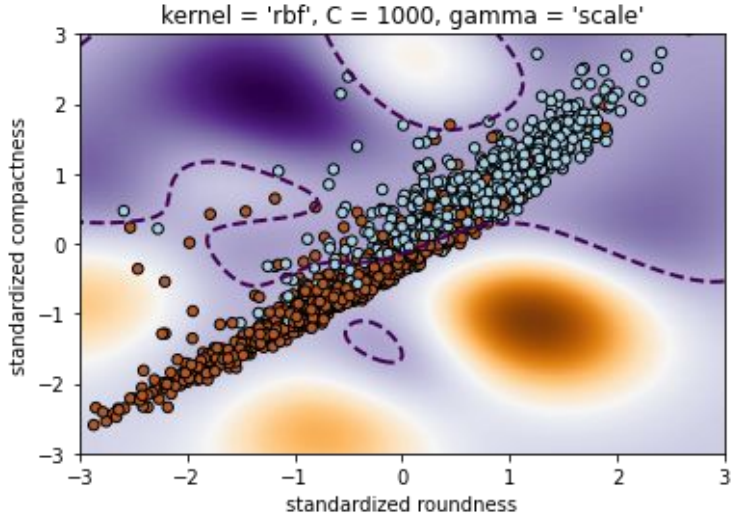

ROC svm.SVC(kernel='poly', degree=3)
True Positive Rate
False Positive Rate
AUC = 0.94


ROC svm.SVC(C=1.0)
True Positive Rate
False Positive Rate
AUC = 0.93

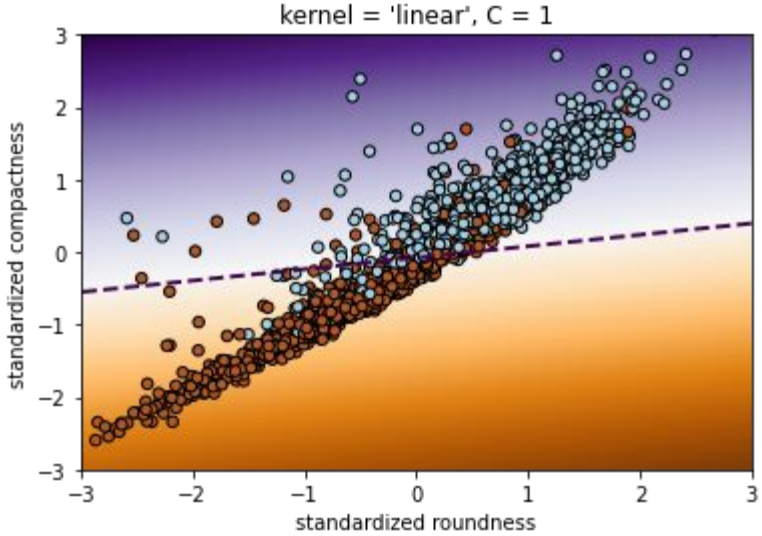

ROC svm.SVC(C=100)
True Positive Rate
False Positive Rate
AUC = 0.95

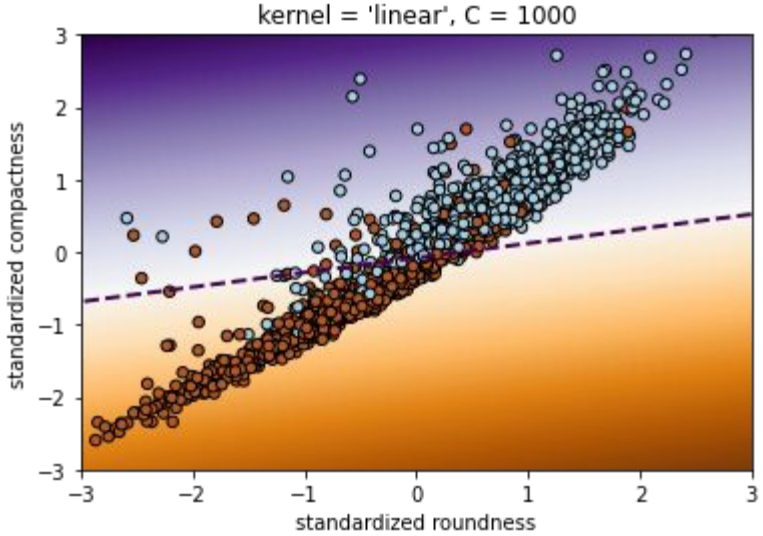
rbf kernel (gaussian)



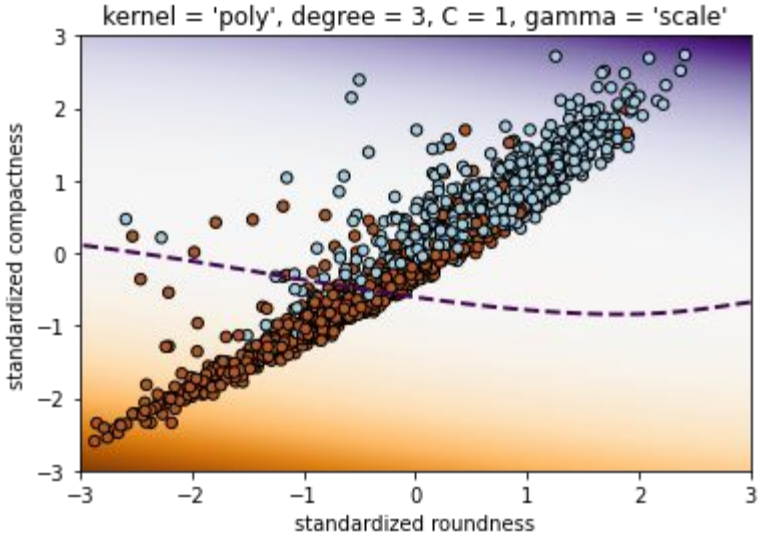


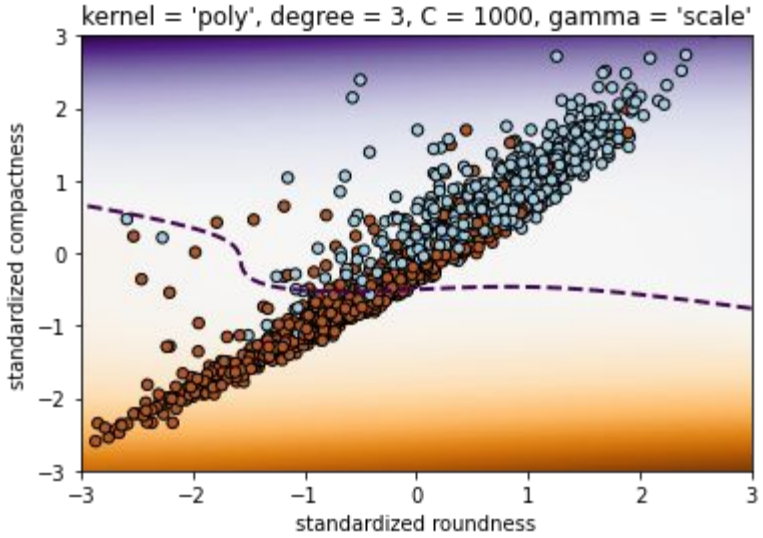
linear kernel



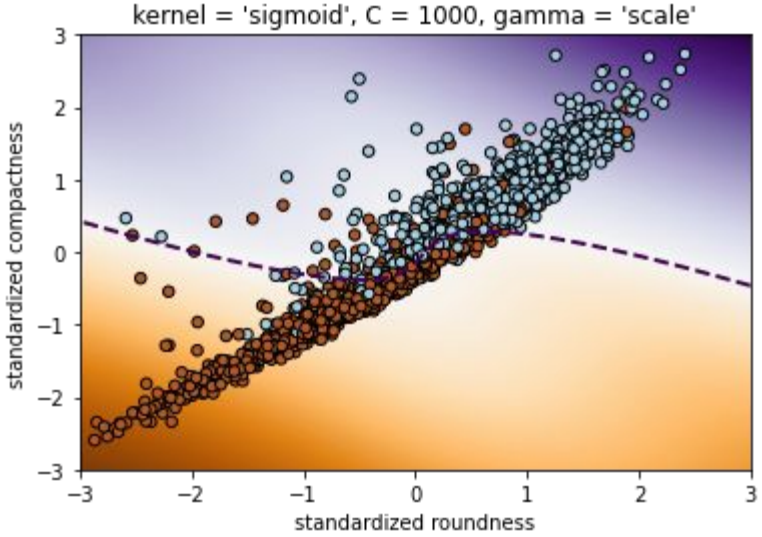


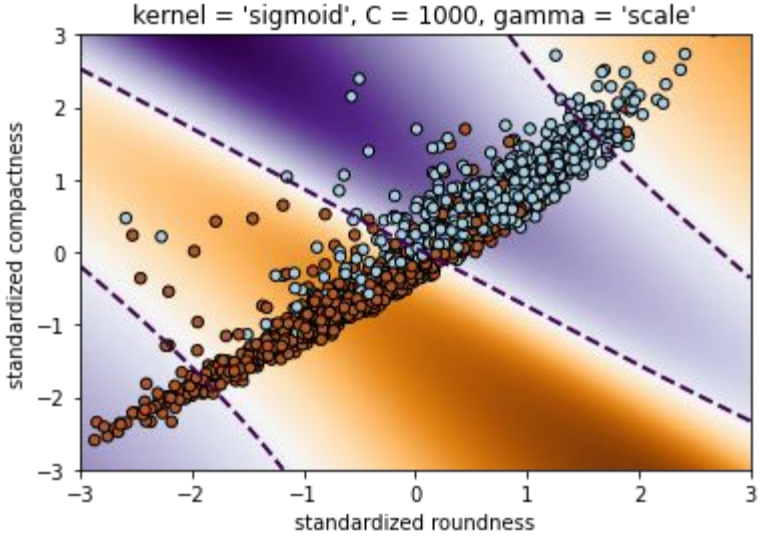
cubic polynomial kernel





sigmoid kernel





References

Dataset from Kaggle
https://www.kaggle.com/datasets/muratkokludataset/pumpkin-seeds-dataset

Research Product
https://github.com/adamwbrew/Research-and-Implementation-of-Machine-Learning-Algorithms

Scikit Learn
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html