



# "Dry Beans"

[About Data set](#)

[correlation Matrix](#)

[Model Building and Enhancing](#)

[Support vector classifier](#)

[Accuracy of SVM](#)

[Decision tree](#)

[Accuracy of decision tree](#)

[Classification Report](#)

[ANN](#)

[Accuracy of ANN](#)

[Summary of ANN model](#)

[visualization ANN](#)

[Summary of Models](#)

## About Data set

Dry beans is a classification data set about: distinguish seven different registered varieties of dry beans with similar features in order to obtain uniform seed classification.

**Total of 16 features; 12 dimensions and 4 shape forms, were obtained from the grains.**

Cleaning and Preparing the data

- Feature Selection
- Dealing with missing data
- Dealing with outliers
- Feature Encoding
- **After drop duplicated:** The shape after drop duplicates is: (13543, 17)

#### Information about data set

Information about DataFrame

```
dry = pd.read_csv("Dry_Bean_Dataset.csv")
print(f"shape of the dataset {dry.shape}")
```

shape of the dataset (13611, 17)

```
dry.head()
```

	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRatio	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	roundness	Compactness	ShapeFactor1	ShapeFactor2	ShapeFactor3
0	28395	610.291	208.178117	173.888747	1.197191	0.549812	28715	190.141097	0.763923	0.988856	0.958027	0.913358	0.007332	0.003147	0.8
1	28734	638.018	200.524796	182.734419	1.097356	0.411785	29172	191.272751	0.783968	0.984986	0.887034	0.953861	0.006979	0.003564	0.9
2	29380	624.110	212.826130	175.931143	1.209713	0.562727	29690	193.410904	0.778113	0.989559	0.947849	0.908774	0.007244	0.003048	0.8
3	30008	645.884	210.557999	182.516516	1.153638	0.498616	30724	195.467062	0.782681	0.976696	0.903936	0.928329	0.007017	0.003215	0.8
4	30140	620.134	201.847882	190.279279	1.060798	0.333680	30417	195.896503	0.773098	0.990893	0.984877	0.970516	0.006697	0.003665	0.9

```
dry.tail()
```

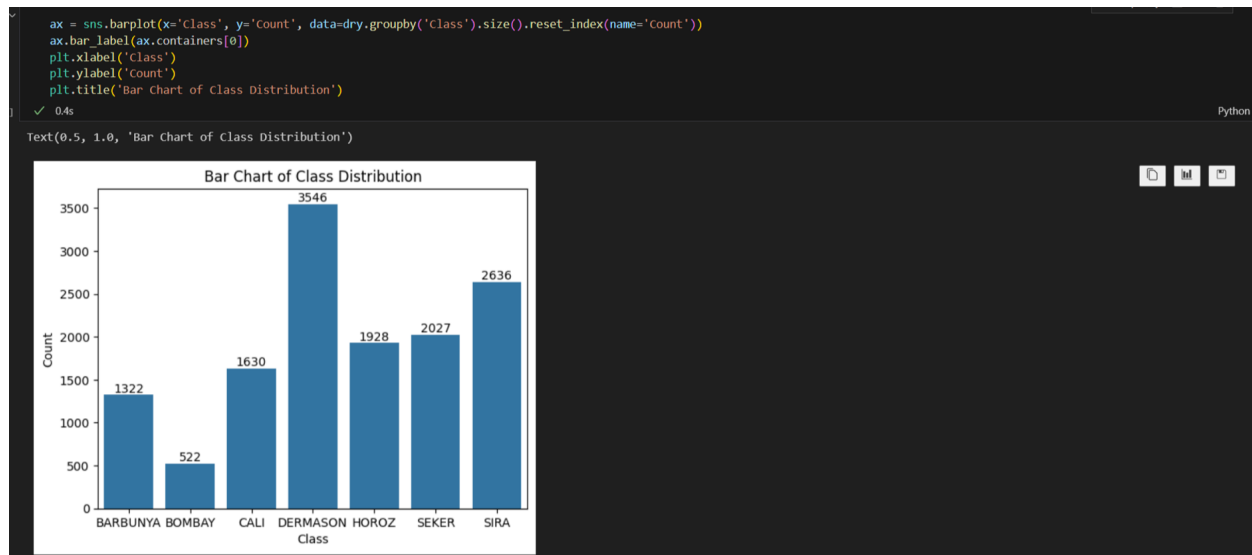
	Area	Perimeter	MajorAxisLength	MinorAxisLength	AspectRatio	Eccentricity	ConvexArea	EquivDiameter	Extent	Solidity	roundness	Compactness	ShapeFactor1	ShapeFactor2	ShapeFactor3
13606	42097	759.696	288.721612	185.944705	1.552728	0.765002	42508	231.515799	0.714574	0.990331	0.916603	0.801865	0.006858	0.001749	0.8
13607	42101	757.499	281.576392	190.713136	1.476439	0.735702	42494	231.526798	0.799943	0.990752	0.922015	0.822252	0.006688	0.001886	0.8
13608	42139	759.321	281.539928	191.187979	1.472582	0.734065	42569	231.631261	0.729932	0.989899	0.918424	0.822730	0.006681	0.001888	0.8

#### Describe data

```
dry.describe(include="o")
```

	Class
count	13611
unique	7
top	DERMASON
freq	3546

#### Graph classes

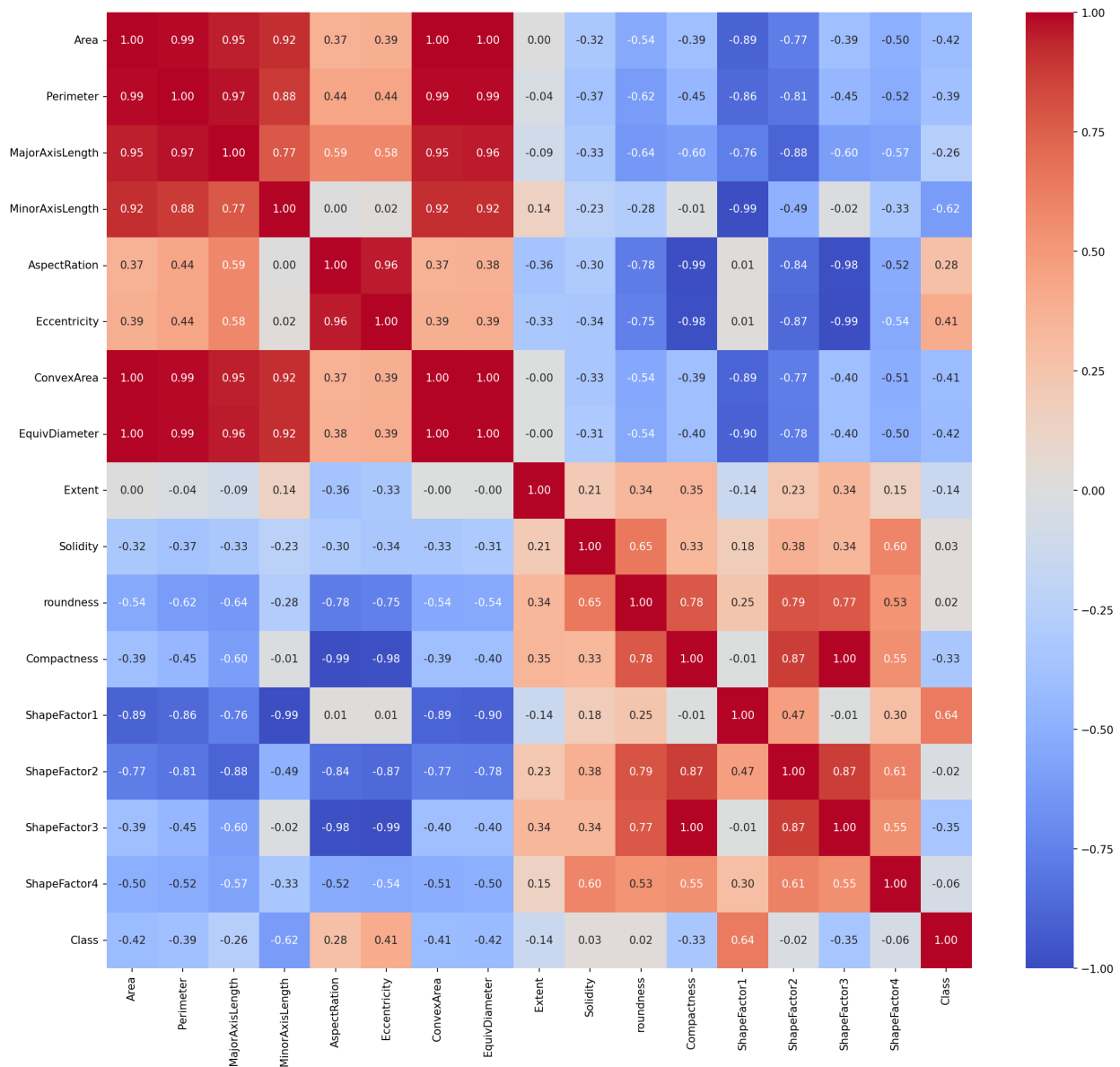


## correlation Matrix

```
dry.select_dtypes(include=['number']).columns
# correlation matrix
corr = dry.corr(numeric_only=True)

# figure settings
plt.figure(figsize=(18,16), dpi=150)

# corr matrix "heatmap"
# sns.heatmap(corr, cmap="Blues")
sns.heatmap(corr, fmt='.2f', annot=True, vmin=-1, center=0, vmax=
```



# Model Building and Enhancing

## 1-Standard scaler

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler(copy=True, with_std=True)
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

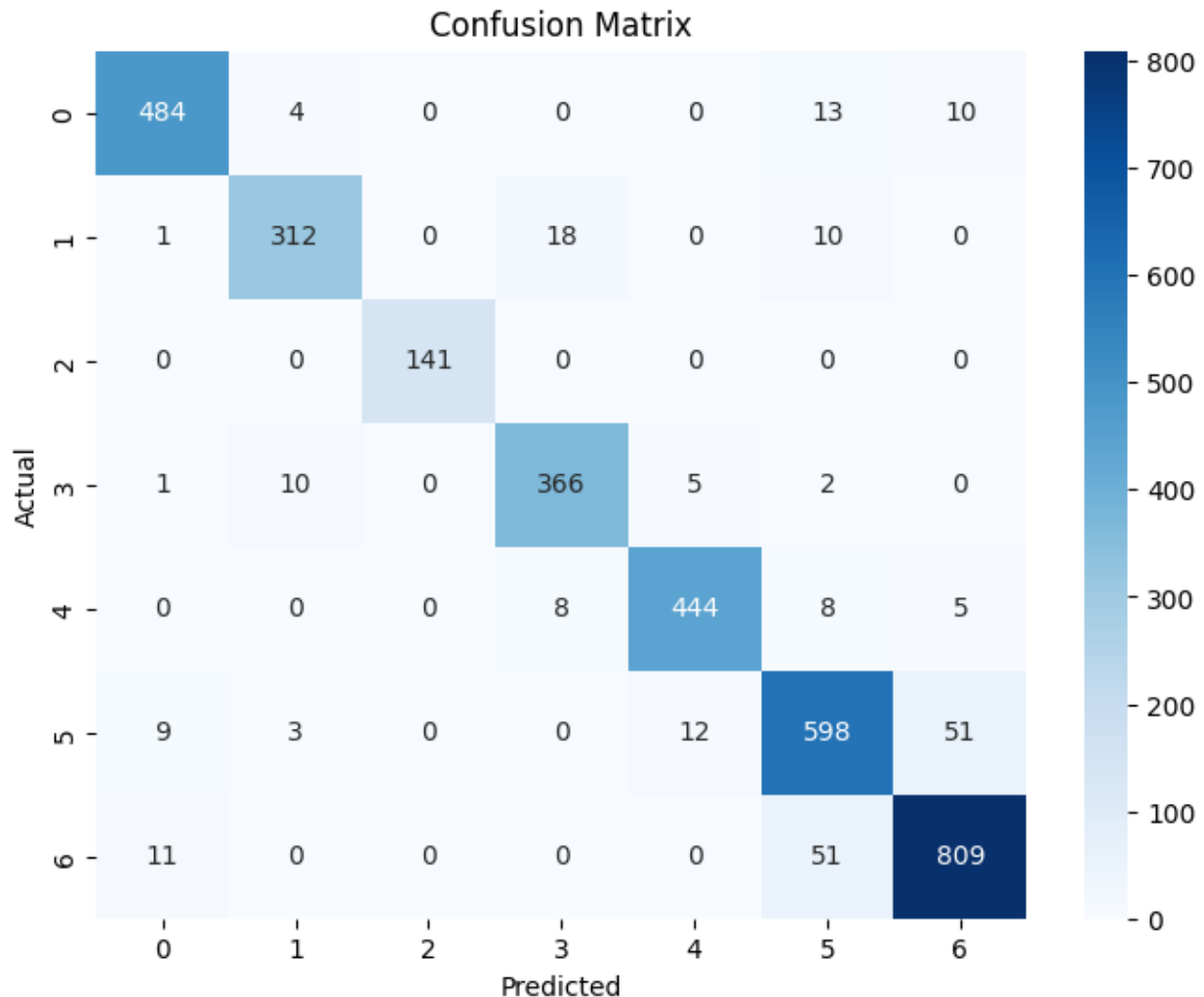
# Support vector classifier

```
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score, KFold

svc = SVC(kernel="rbf", C=1)
svc.fit(X_train, y_train)

cv = KFold(n_splits=5, shuffle=True, random_state=42)
scores_svc = cross_val_score(svc, X_train, y_train, cv=cv)
print("Cross-validated scores:", scores_svc)
print(f"Mean Cross-validated-accuracy: {scores_svc.mean()}")
```

## confusion matrix for SVM



## Accuracy of SVM

train score: 0.9318696465491779

test score: 0.9314825753101004

## Decision tree

```
from sklearn.tree import DecisionTreeClassifier
# Create a decision tree classifier with customized hyperparameters
tree_clf = DecisionTreeClassifier(
    max_depth=550, # Maximum depth of the tree
    min_samples_split=350, # Minimum number of samples required
```

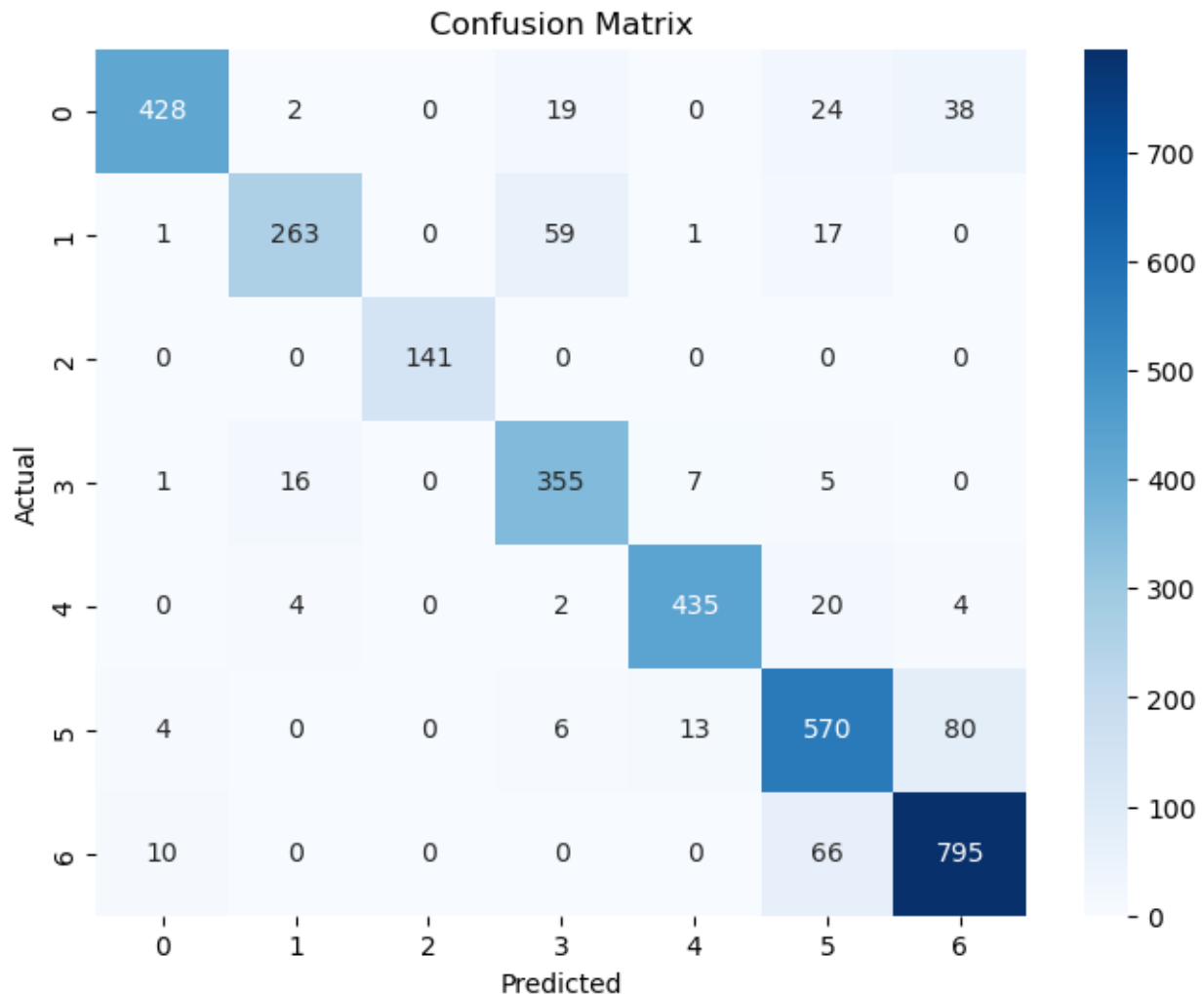
```

min_samples_leaf=150, # Minimum number of samples required
random_state=42 # Random state for reproducibility
)
# Train the classifier on the training data
tree_clf.fit(X_train, y_train)

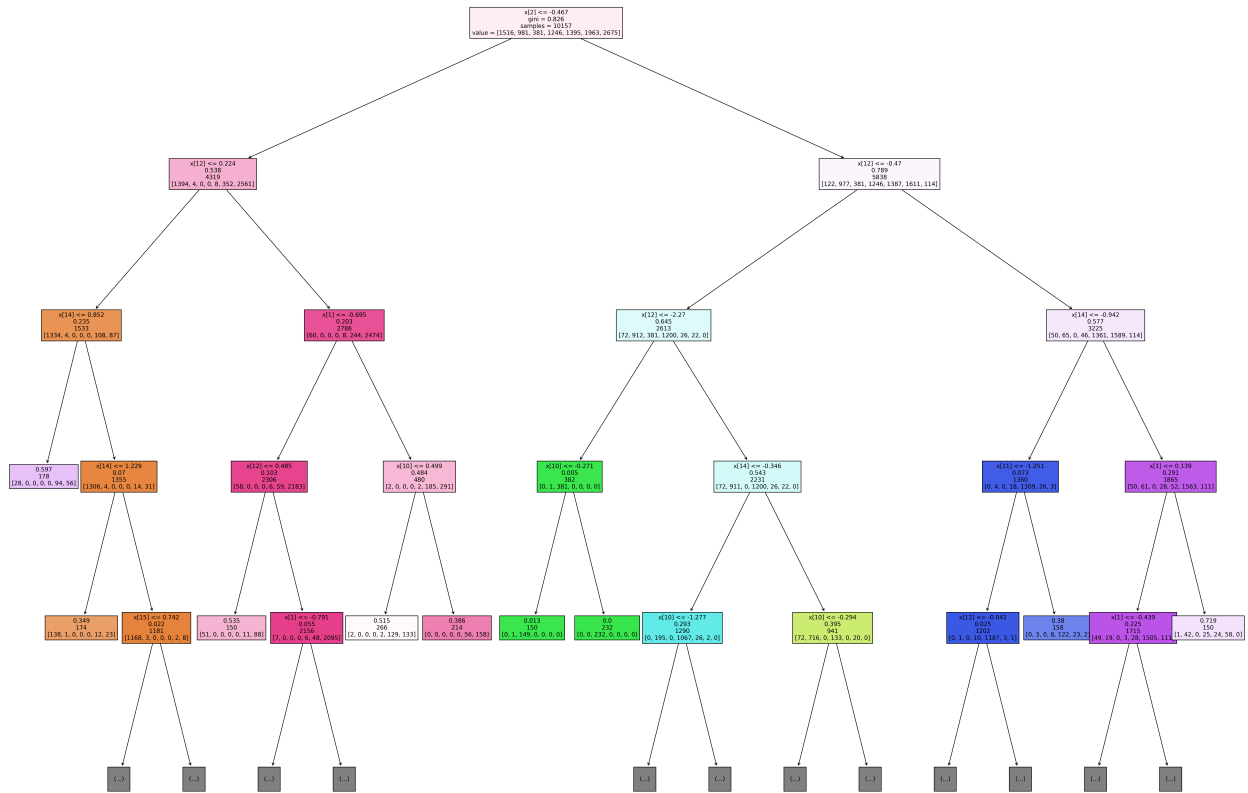
cv = KFold(n_splits=5, shuffle=True, random_state=42)
scores_tree = cross_val_score(tree_clf, X_train, y_train, cv=cv)
print("Cross-validated scores:", scores_tree)
print(f"Mean Cross-validated-accuracy: {scores_tree.mean()}")

```

### Confusion matrix for Decision tree



## visualization of decision tree



## Accuracy of decision tree

train score: 0.8885497686324703

test score: 0.8821618428824571

## Classification Report

	precision	recall	f1-score	support
0	0.96	0.84	0.90	511
1	0.92	0.77	0.84	341
2	1.00	1.00	1.00	141



3	0.80	0.92	0.86	384
4	0.95	0.94	0.94	465
5	0.81	0.85	0.83	673
6	0.87	0.91	0.89	871
accuracy			0.88	3386
macro avg	0.90	0.89	0.89	3386
weighted avg	0.89	0.88	0.88	3386

## ANN

```
import tensorflow as tf
from tensorflow import keras
model = keras.Sequential([
    keras.layers.Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    keras.layers.Dense(110, activation='relu'),
    keras.layers.Dense(110, activation='relu'),
    keras.layers.Dense(7, activation='softmax')# # Number of unique classes
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy', # Use 'sparse_categorical_crossentropy'
              metrics=['accuracy'])
```

## Accuracy of ANN

```
history = model.fit(X_train, y_train, epochs=10, batch_size=32,
                   validation_data=(X_test, y_test))
test_loss, test_accuracy = model.evaluate(X_test, y_test)

# Evaluate the model
print('test Loss:', test_loss)
print('test Accuracy:', test_accuracy)
```

test Loss: 0.2116522192955017

test Accuracy: 0.9220318794250488

```
Epoch 1/10
301/318 — 0s 838us/step - accuracy: 0.7906 - loss: 0.6362
318/318 — 2s 2ms/step - accuracy: 0.7952 - loss: 0.6216 - val_accuracy: 0.9155 - val_loss: 0.2345
Epoch 2/10
318/318 — 0s 1ms/step - accuracy: 0.9222 - loss: 0.2152 - val_accuracy: 0.9279 - val_loss: 0.2157
Epoch 3/10
318/318 — 0s 1ms/step - accuracy: 0.9214 - loss: 0.2076 - val_accuracy: 0.9229 - val_loss: 0.2202
Epoch 4/10
318/318 — 1s 1ms/step - accuracy: 0.9260 - loss: 0.2006 - val_accuracy: 0.9247 - val_loss: 0.2091
Epoch 5/10
318/318 — 0s 1ms/step - accuracy: 0.9299 - loss: 0.1903 - val_accuracy: 0.9182 - val_loss: 0.2158
Epoch 6/10
318/318 — 0s 1ms/step - accuracy: 0.9297 - loss: 0.1883 - val_accuracy: 0.9262 - val_loss: 0.2076
Epoch 7/10
318/318 — 0s 1ms/step - accuracy: 0.9289 - loss: 0.1881 - val_accuracy: 0.9259 - val_loss: 0.2034
Epoch 8/10
318/318 — 0s 1ms/step - accuracy: 0.9348 - loss: 0.1712 - val_accuracy: 0.9149 - val_loss: 0.2254
Epoch 9/10
318/318 — 0s 1ms/step - accuracy: 0.9296 - loss: 0.1801 - val_accuracy: 0.9247 - val_loss: 0.2109
Epoch 10/10
318/318 — 0s 1ms/step - accuracy: 0.9336 - loss: 0.1794 - val_accuracy: 0.9220 - val_loss: 0.2117
106/106 — 0s 784us/step - accuracy: 0.9317 - loss: 0.2029
test loss: 0.2116522192955017
test Accuracy: 0.9220318794250488
```

## Summary of ANN model

```
model.summary()
[70] ✓ 0.0s

...
Model: "sequential"

...


| Layer (type)    | Output Shape | Param # |
|-----------------|--------------|---------|
| dense (Dense)   | (None, 128)  | 2,176   |
| dense_1 (Dense) | (None, 110)  | 14,190  |
| dense_2 (Dense) | (None, 110)  | 12,210  |
| dense_3 (Dense) | (None, 7)    | 777     |



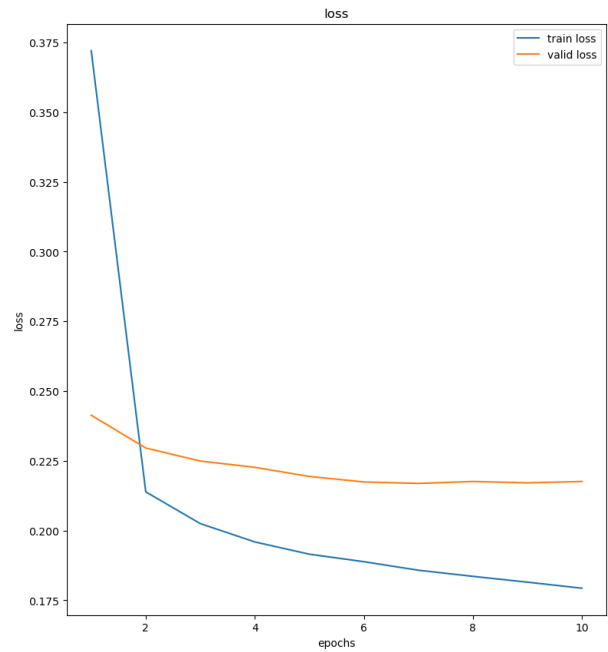
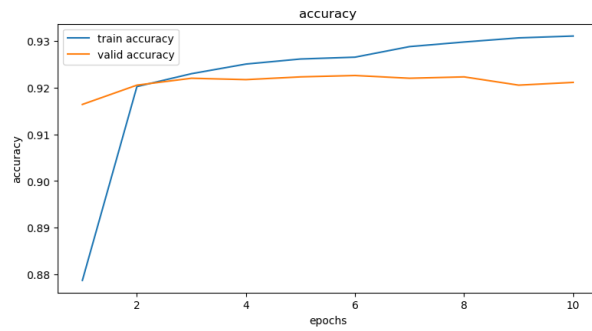
...
Total params: 88,061 (343.99 KB)

...
Trainable params: 29,353 (114.66 KB)

...
Non-trainable params: 0 (0.00 B)

...
Optimizer params: 58,708 (229.33 KB)
```

## visualization ANN



## Summary of Models

SVC	Decision Tree	ANN
train score: 0.9318696465491779	train score: 0.8885497686324703	test Loss: 0.2116522192955017
test score: 0.9314825753101004	test score: 0.8821618428824571	test score: 0.9220318794250488