PROJECT REPORT CMPS 470/570

Team: Tech19

Team members: Drew Hutchinson, Zichuo Wang

Instructor: Dr. Ömer Soysal

Report structure

- Description of the project
- Description of the raw data
- Preprocessing
- Feature extraction
- Description of the feature data
- Description of the model
- Performance of the model
- Team members & roles / Task completion report

Description of the project

Develop a machine learning application using Python and related libraries,

to solve the **penguin species classification** problem

with the dataset **Palmer Penguins**.

This application can:

Pre-process data,

Create ML models based on the methods of K-NN, ANN, SVM and DT.

Display and export results and model parameters.

Description of the raw data

Data source: Palmer Penguins

https://github.com/allisonhorst/palmerpenguins

Attributes and their types

Attributes	SID	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
Types	Numerical	Texts	Texts	Numerical	Numerical	Numerical	Numerical	Texts	Numerical
Sample	2	Adelie	Torgersen	39.5	17.4	186	3800	female	2007

cmpsML_Tech19\OTHER\penguins.csv

cmpsML_Tech19\OTHER\data_description.xlsx\raw_data_description

Discussion:

We downloaded the file penguins.csv as our original dataset.

It has 9 columns and 344 rows. The column 'species' is the target column.

Preprocessing

We imported the original data set from penguins.csv.

We defined two functions handling the missing values.

We also applied scalers to the dropped missing values data set.

```
"<preprocessing>" module begins.
Original data set D0:
This Data Set has 344 rows, 9 columns
   SID species     island  ...  body_mass_g     sex  year
0    1  Adelie  Torgersen  ...       3750.0    male  2007
1    2  Adelie  Torgersen  ...       3800.0  female  2007
2    3  Adelie  Torgersen  ...       3250.0  female  2007
3    4  Adelie  Torgersen  ...          NaN     NaN  2007
4    5  Adelie  Torgersen  ...       3450.0  female  2007
```

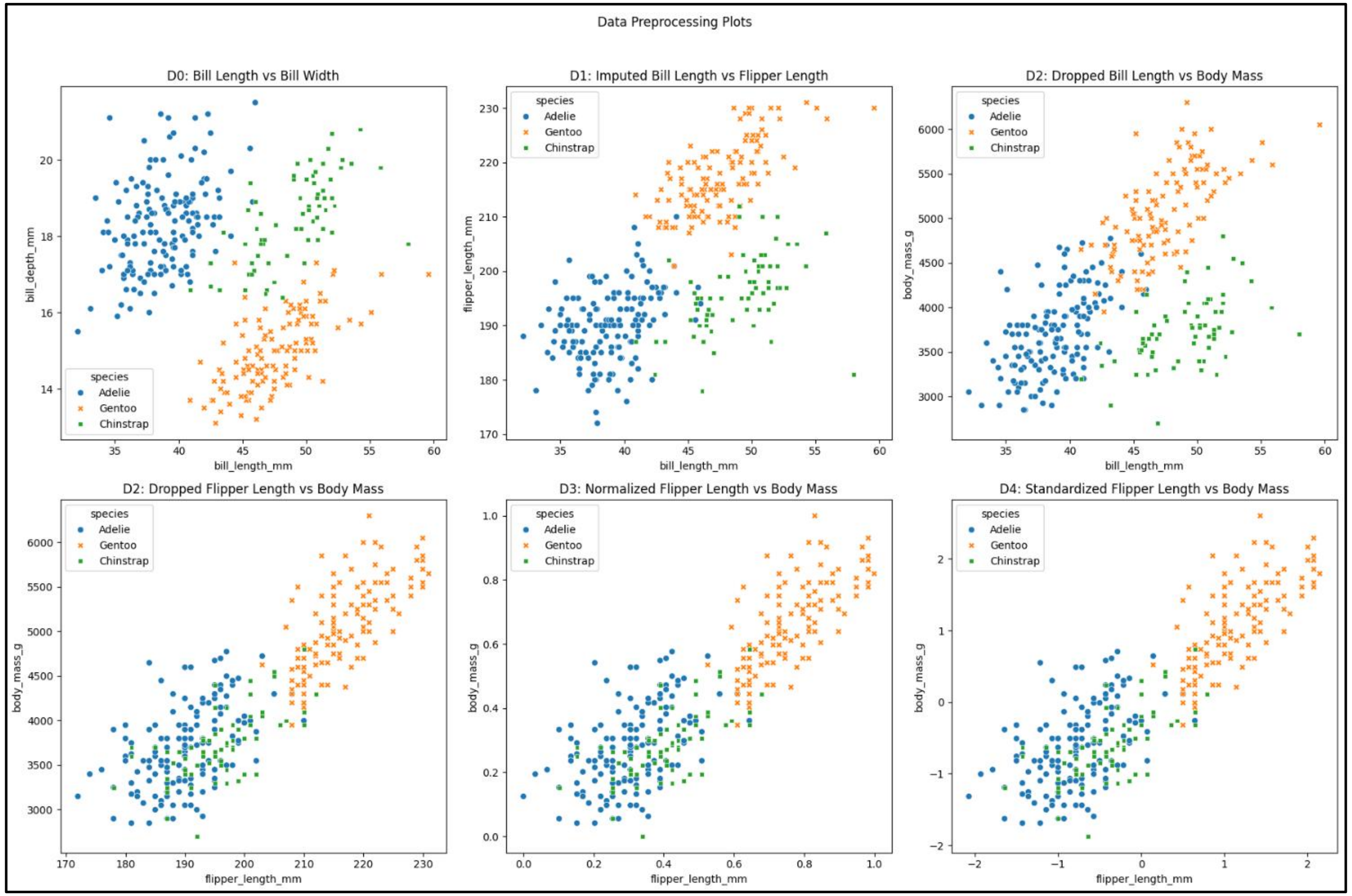
```
Imputed missing values data set D1:
This Data Set has 344 rows, 9 columns
   SID species     island  ...  body_mass_g     sex  year
0    1  Adelie  Torgersen  ...  3750.000000    male  2007
1    2  Adelie  Torgersen  ...  3800.000000  female  2007
2    3  Adelie  Torgersen  ...  3250.000000  female  2007
3    4  Adelie  Torgersen  ...  4201.754386    male  2007
4    5  Adelie  Torgersen  ...  3450.000000  female  2007

[5 rows x 9 columns]
Dropped missing values data set D2:
This Data Set has 333 rows, 9 columns
   SID species     island  ...  body_mass_g     sex  year
0    1  Adelie  Torgersen  ...       3750.0    male  2007
1    2  Adelie  Torgersen  ...       3800.0  female  2007
2    3  Adelie  Torgersen  ...       3250.0  female  2007
4    5  Adelie  Torgersen  ...       3450.0  female  2007
5    6  Adelie  Torgersen  ...       3650.0    male  2007
```

```
Normalized data set D3:
This Data Set has 333 rows, 9 columns
   SID species     island  ...  body_mass_g     sex  year
0    1  Adelie  Torgersen  ...     0.291667    male  2007
1    2  Adelie  Torgersen  ...     0.305556  female  2007
2    3  Adelie  Torgersen  ...     0.152778  female  2007
4    5  Adelie  Torgersen  ...     0.208333  female  2007
5    6  Adelie  Torgersen  ...     0.263889    male  2007

[5 rows x 9 columns]
Standardized data set D4:
This Data Set has 333 rows, 9 columns
   SID species     island  ...  body_mass_g     sex  year
0    1  Adelie  Torgersen  ...    -0.568475    male  2007
1    2  Adelie  Torgersen  ...    -0.506286  female  2007
2    3  Adelie  Torgersen  ...    -1.190361  female  2007
4    5  Adelie  Torgersen  ...    -0.941606  female  2007
5    6  Adelie  Torgersen  ...    -0.692852    male  2007
```

cmpsML_Tech19\CODE\OUTPUT\preprocessed_data.xlsx

Preprocessing

Here is our data preprocessing plots:



cmpsML_Tech19\CODE\OUTPUT\dataset plots\Tech19_PA1_Plots.png

Discussion:

In this part we tried to read files from the folder, apply scalers to the dataset and generate plots.

In the second row of the plots, we compared flipper length vs body mass from 3 datasets: Dropped missing value dataset D2, Normalized dataset D3, Standardlized dataset D4.

You can see that **scaler won't change the distribution of the data**.

Feature extraction

We dropped the rows where have missing values.
We dropped the irrelevant column('SID' , 'Year').
We applied one-hot encoding for categorical columns('sex', 'island') and label encoding for species column.

```python
def preprocessing_penguins(df):  1 usage
    df = df.copy()
    df = drop_missing_values(df)
    df = drop_irrelevant_columns(df)
    df = one_hot_encode(df)
    df = label_encode_species(df)
    return df
```

```
"<FeatureExtraction>" module begins.
Original dataset D0:
This Data Set has 344 rows, 9 columns
   SID species     island  ...  body_mass_g     sex  year
0    1  Adelie  Torgersen  ...       3750.0    male  2007
1    2  Adelie  Torgersen  ...       3800.0  female  2007
2    3  Adelie  Torgersen  ...       3250.0  female  2007
3    4  Adelie  Torgersen  ...          NaN     NaN  2007
4    5  Adelie  Torgersen  ...       3450.0  female  2007

[5 rows x 9 columns]
Preprocessed dataset D1:
This Data Set has 333 rows, 9 columns
   species  bill_length_mm  ...  island_Dream  island_Torgersen
0        0            39.1  ...             0                 1
1        0            39.5  ...             0                 1
2        0            40.3  ...             0                 1
4        0            36.7  ...             0                 1
5        0            39.3  ...             0                 1

[5 rows x 9 columns]
```

Feature extraction

We divided the dataset into 3 sub-sets (train, val, test = 0.6, 0.2, 0.2).

We added some noise to the training dataset. Different noise_std were applied.

```
def split_data(df):   1 usage
    train_ratio = 0.6
    val_ratio = 0.2
    test_ratio = 0.2
    random_state = 42
    df = df.copy()
    train_val, test = train_test_split( *arrays: df, test_size=test_ratio, random_state=random_state)
    train,val = train_test_split( *arrays: train_val, test_size=val_ratio/(train_ratio + val_ratio),

    return train,val,test
```

```
#dataset splitting
train,val,test = split_data(D1)
```

```
#add noise to the training dataset
train_noise = noisify(train)

#applying scaler to sub-sets
train_std,val_std,test_std = standardize_data(train_noise,val,test)
```

```
def noisify(df):   1 usage
    noise_std_dict = {
        'bill_length_mm': 3.0,
        'bill_depth_mm': 1.5,
        'flipper_length_mm': 7.0,
        'body_mass_g': 150.0
    }

    df = df.copy()

    for col, std in noise_std_dict.items():
        if col in df.columns:
            noise = np.random.normal(loc=0, scale=std, size=df[col].shape)
            df[col] += noise
        else:
            print(f"Column '{col}' not found in the dataset.")
```

cmpsML_Tech19\CODE\OUTPUT\feature_extracted_data.xlsx

Feature extraction

We extracted the sub-sets to excel files as requested.

cmpsML_Tech19\CODE\OUTPUT\feature_extracted_data.xlsx

cmpsML_Tech19\CODE\INPUT\TEST\val_test_std.xlsx

cmpsML_Tech19\CODE\INPUT\TRAIN\train_std.xlsx

```
def standardize_data(df1,df2,df3):  1 usage
    df1 = df1.copy()
    df2 = df2.copy()
    df3 = df3.copy()
    numeric_cols = ['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g']
    scaler = StandardScaler()
    cols_to_scale = [col for col in numeric_cols if col in df1.columns]
    df1[cols_to_scale] = scaler.fit_transform(df1[cols_to_scale])
    df2[cols_to_scale] = scaler.transform(df2[cols_to_scale])
    df3[cols_to_scale] = scaler.transform(df3[cols_to_scale])
    return df1,df2,df3
```

Discussion:

In PA2 we made a mistake that applying std-scaler **before** splitting the dataset.

This will lead to **data leakage**. When we scale based on the entire dataset, the scaling parameters (mean and standard deviation) are influenced by the validation and test data. This means that **information from the validation and test sets leaks into the training process**. Also we should only apply ‘**fit_transform**’ to training dataset, and ‘**transform**’ to val and test datasets.

We fixed this issue after discovering it. Split the dataset into train, val and test first then scale the sub-sets seperately.

Description of the feature data

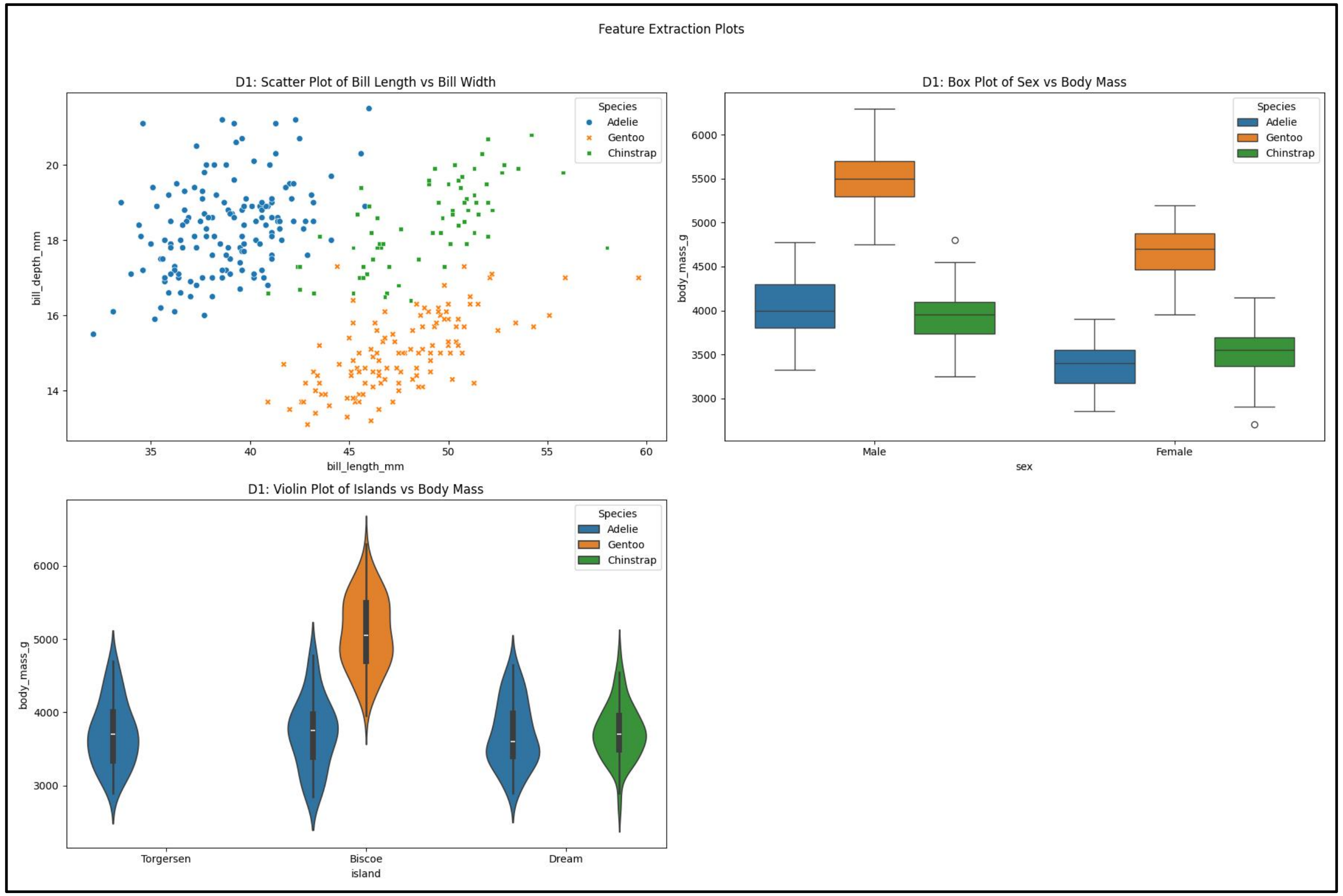
We have 9 numerical columns after feature extraction:

Features	Type	Notes
species	Numerical	Target, label encoding, 0 = 'Adelie', 1 = 'Chinstrap', 2 = 'Gentoo'
bill_length_mm	Numerical	Standardized
bill_depth_mm	Numerical	Standardized
flipper_length_mm	Numerical	Standardized
body_mass_g	Numerical	Standardized
sex_male	Numerical	one-hot encoding, 1 = 'male', 0 = 'female'
island_Biscoe	Numerical	one-hot encoding, 1 = True, 0 = False
island_Dream	Numerical	one-hot encoding, 1 = True, 0 = False
island_Torgersen	Numerical	one-hot encoding, 1 = True, 0 = False

cmpsML_Tech19\OTHER\data_description.xlsx\feature_data_description

Description of the feature data

Here is a feature plot of original dataset:



cmpsML_Tech19\CODE\OUTPUT\dataset plots\Tech19_PA2_Plots.png

Discussion:

From the first plot on top, left, we can see that these three penguin species **do not have much overlap** in bill length and width distribution, which means **they are seperable in this feature**. (This is why we added some **noise** later.)

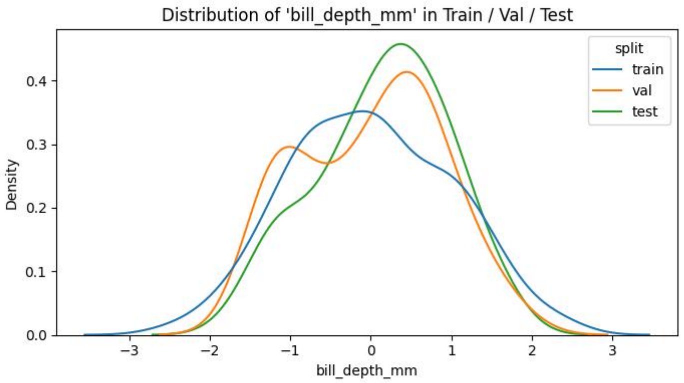
From the second plot on top, right, we can see that **Gentoo is much heavier** than the other two species.

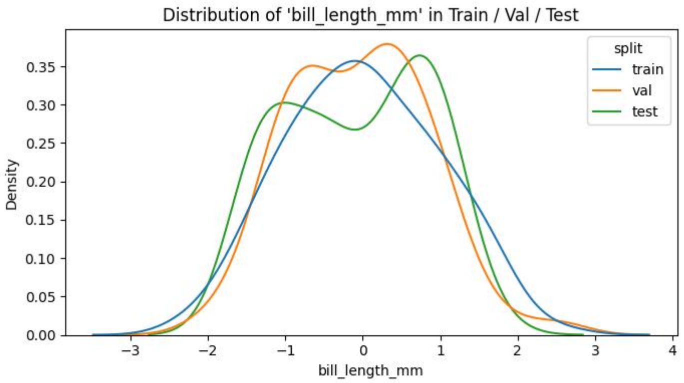
From the third plot on bottom, left, we can see that Gentoo only lives on island Biscoe.
Chinstrap only lives on island Dream.
On island Torgersen, there is only one specie - Adelie.
This makes **island and body mass good seperation nodes in DT**.

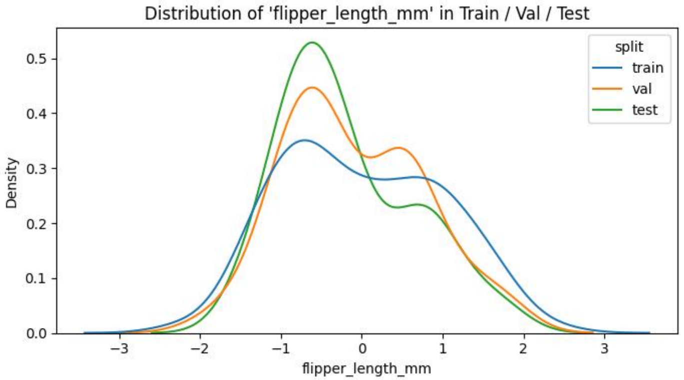
The feature island highly correlates to species.
We decided to drop the 'island' column during the training.

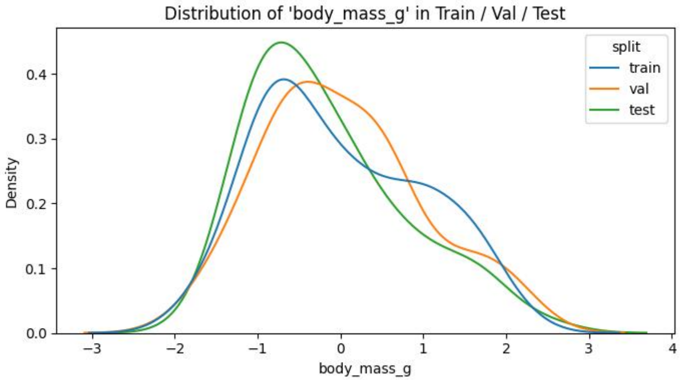
Description of the feature data

Here are our distribution of train/val/test datasets.





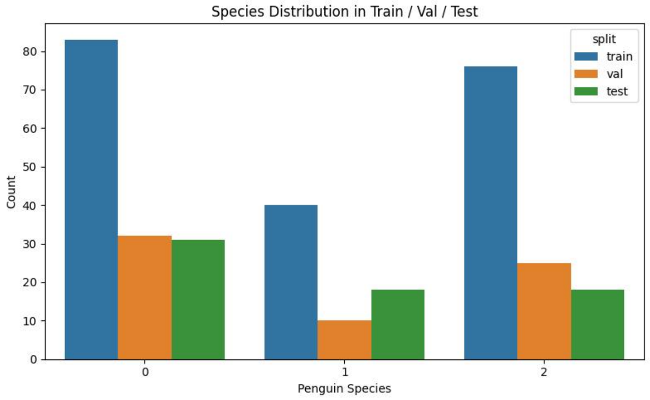




Discussion:

The distribution looks similar in val/test datasets,
and a little bit different in training dataset
due to the noise we added.

Basically our operation is reasonable.



cmpsML_Tech19\CODE\OUTPUT\dataset plots\

Description of the model

We built 5 models: KNN, DT, SVM, Grid search ANN and Random setting ANN.

```
'KNN': KNeighborsClassifier(
    n_neighbors=5,
    weights='distance',
    p=2
),

'DT': DecisionTreeClassifier(
    criterion='gini',
    max_depth=4,
    min_samples_split=5,
    random_state=42
),

'SVM': SVC(
    C=1.0,
    kernel='rbf',
    gamma='scale',
    probability=True,
    random_state=42
)
```

Model parameters description

KNN：

n_neighbours, k = 5

weights, weight function for neighbours (uniform/distance)

p, 1 = L1 distance, 2 = L2 distance

DT:

criterion, function to measure the quality of a split (gini/entropy)

min_samples_split, minimum number of samples required to split

SVM:

c, regularization parameter, default setting is 1.0.

Description of the model

```
'RANDOM_ANN': MLPClassifier(
    solver='adam',
    batch_size=60,
    max_iter=50,
    early_stopping=True,
    random_state=42,
    hidden_layer_sizes=(6,3),
    activation = 'relu',
    alpha = 0.001,
    learning_rate_init = 0.05,
    validation_fraction=0.2
),
```

Model parameters description

ANN：

solver, the solver for weight optimization

batch_size = 60, size of minibatches for optimizers

max_iter, maximum number of iterations

early_stopping = True, stop when validation score is not improving

hidden_layer_size = (6,3) , 6 nodes at first layer and 3 nodes at second layer

activation, the activation function of the network

alpha, L2 regularization

learning_rate_init, the initial learning rate

validation_fraction, the proportion of training data to set aside as validation set for early stopping

Description of the model

```
def grid_search_ann(X, y):  1 usage
    print("\nRunning GridSearch for ANN...")
    # define pipeline and hyperparameter grid
    ann_pipeline = Pipeline([
        ("scaler", StandardScaler()),
        ("clf", MLPClassifier(
            solver='adam',
            batch_size='auto',
            max_iter=300,
            early_stopping=False,
            random_state=42
        ))
    ])

    param_grid = {
        "clf__hidden_layer_sizes": [(8, 4), (6, 3),(4,2)],
        "clf__activation": ['relu', 'tanh'],
        "clf__alpha": [0.0001, 0.001],
        "clf__learning_rate_init": [0.001, 0.005, 0.01]
    }

    # Stratified K-Fold cross-validation
    skf = StratifiedKFold(n_splits=3, shuffle=True, random_state=42)
    grid_search = GridSearchCV(
        estimator=ann_pipeline,
        param_grid=param_grid,
        cv=skf,
        scoring='accuracy',
        n_jobs=-1,
        verbose=1
    )
    grid_search.fit(X, y)
```

Grid search was applied to the grid_search_ann.

3-fold cross validation was also applied.

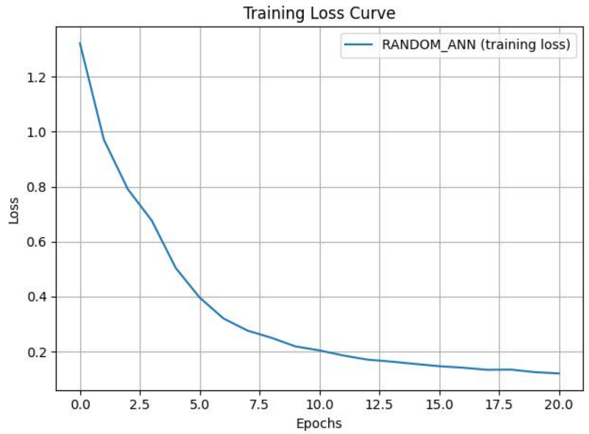
The rest models used 3-tier testing scheme.

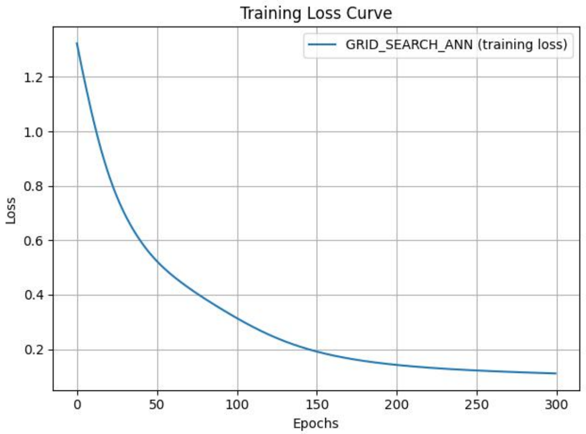
We intentionally set the ANN to be a small one.

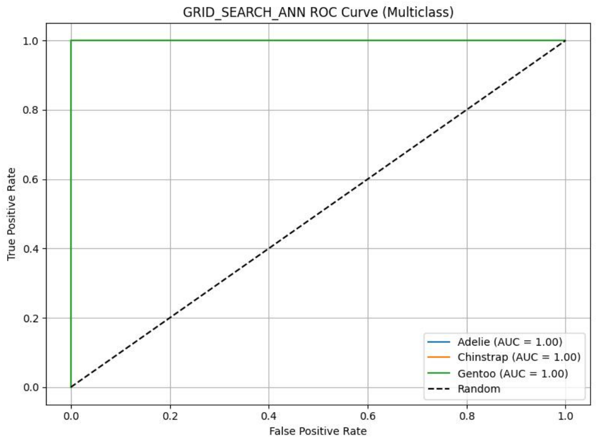
Performance of the model

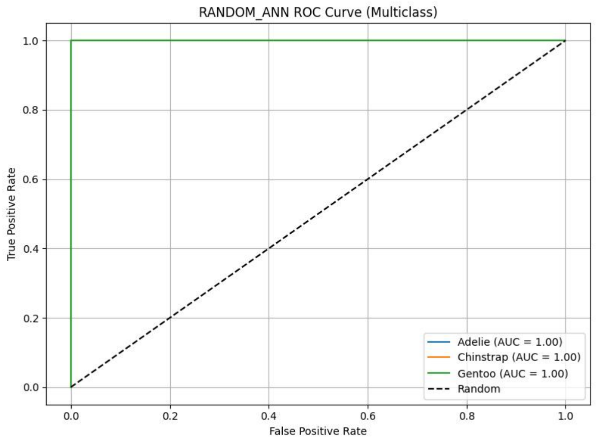
All the plots and excel files are under the folder:

cmpsML_Tech19\CODE\OUTPUT\model performance









Grid search K-fold cross validation results：

A	B
Fold	Accuracy
1	0.955224
2	0.939394
3	0.939394
Mean	0.944671

Training loss curve

Roc curve

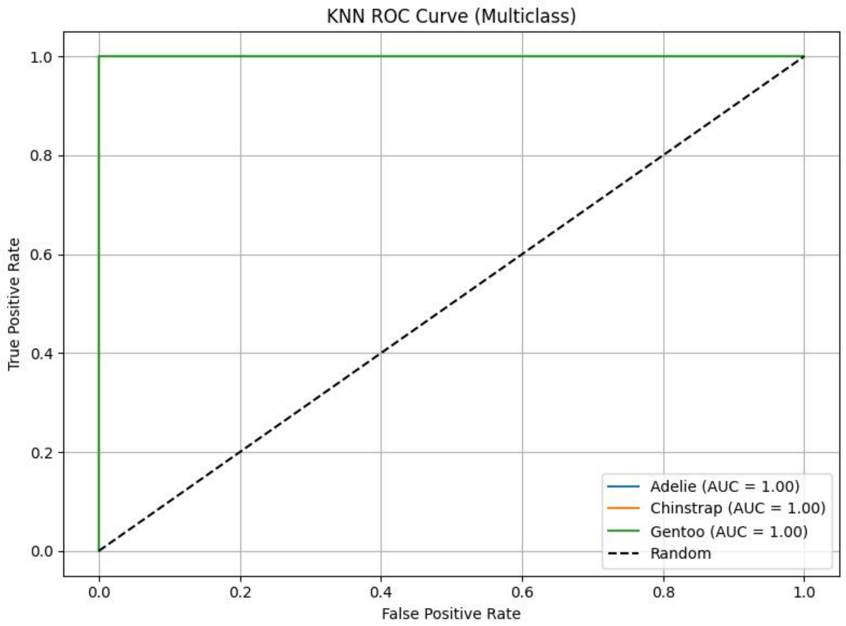
for ANN

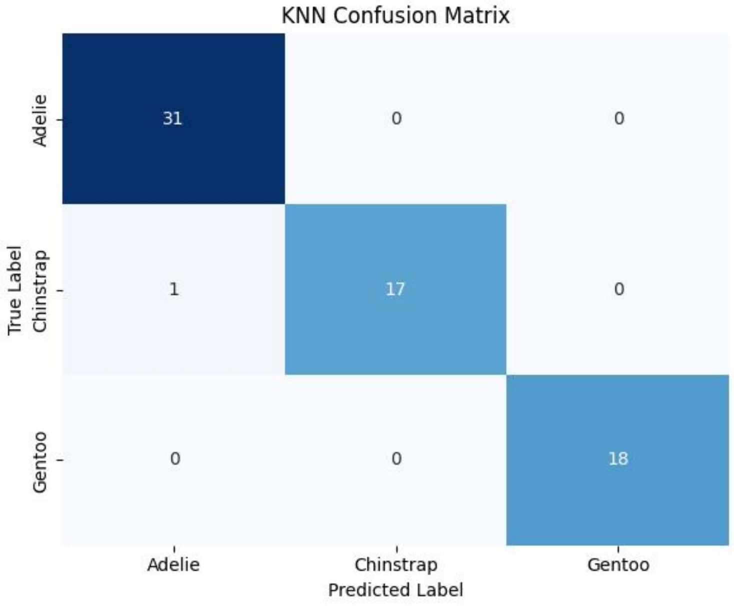
(MLPClassifier doesn’t provide epoch-error curve for val)

Performance of the model

Performance plots for KNN

```
KNN Confusion Matrix:
           Adelie  Chinstrap  Gentoo
Adelie         31          0       0
Chinstrap       1         17       0
Gentoo          0          0      18
KNN Average ROC AUC: 1.0000

KNN Evaluation Metrics:
Accuracy:  0.9851
Precision: 0.9896
Recall:    0.9815
Specificity (avg): 0.9907
F1 Score:  0.9852
```

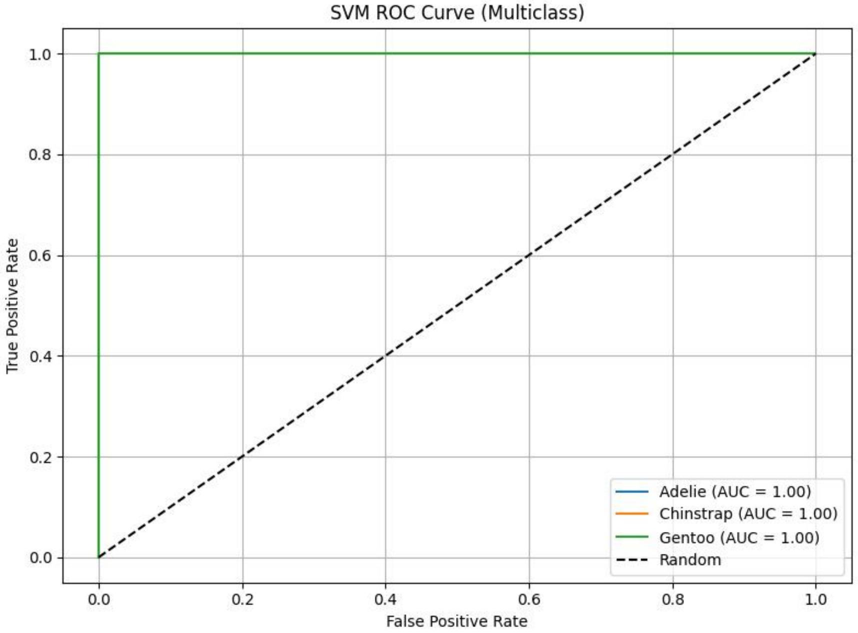


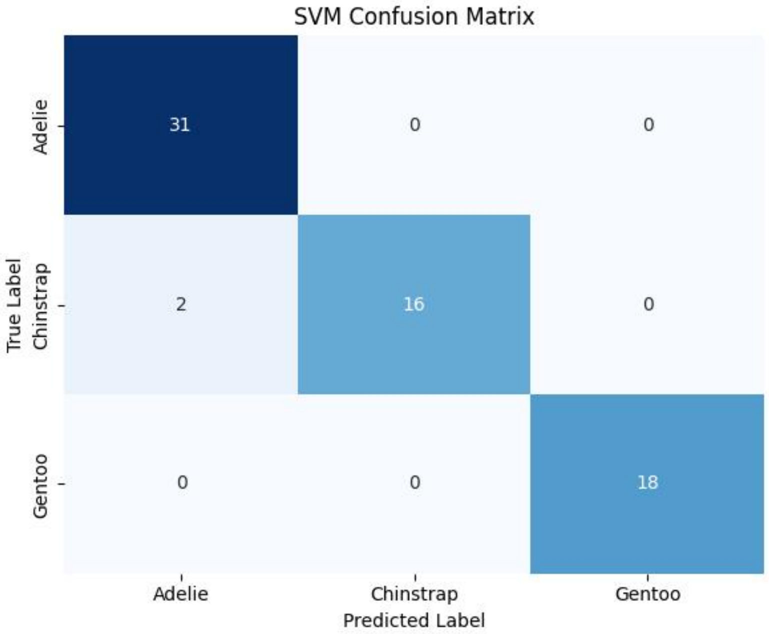


Performance of the model

Performance plots for SVM

```
SVM Confusion Matrix:
           Adelie  Chinstrap  Gentoo
Adelie         31          0       0
Chinstrap       2         16       0
Gentoo          0          0      18
SVM Average ROC AUC: 1.0000

SVM Evaluation Metrics:
Accuracy:  0.9701
Precision: 0.9798
Recall:    0.9630
Specificity (avg): 0.9815
F1 Score:  0.9700
```

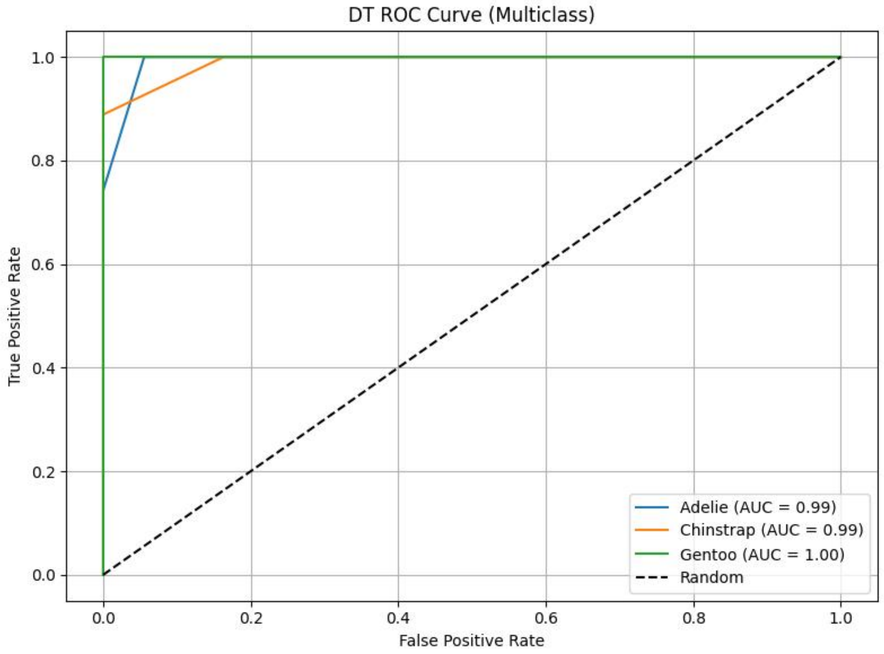


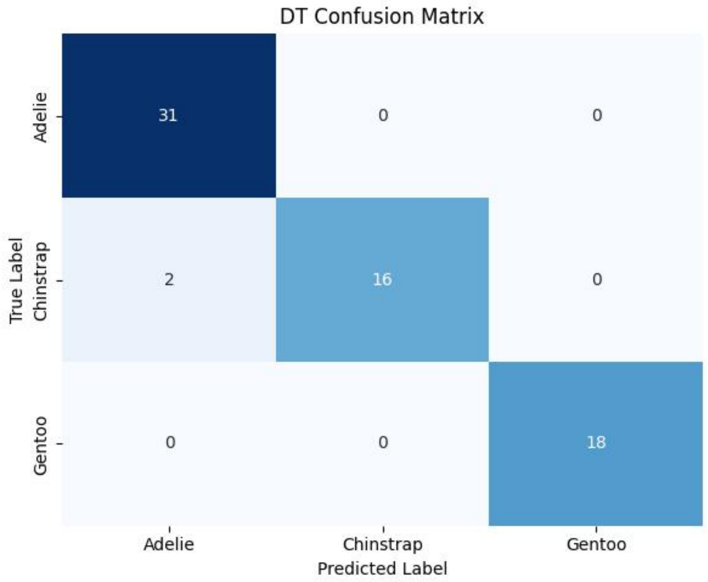


Performance of the model

Performance plots for DT

```
DT Confusion Matrix:
            Adelie  Chinstrap  Gentoo
Adelie          31          0       0
Chinstrap        2         16       0
Gentoo           0          0      18
DT Average ROC AUC: 0.9946

DT Evaluation Metrics:
Accuracy:  0.9701
Precision: 0.9798
Recall:    0.9630
Specificity (avg): 0.9815
F1 Score:  0.9700
```





Performance of the model

Performance plots for RANDOM_ANN

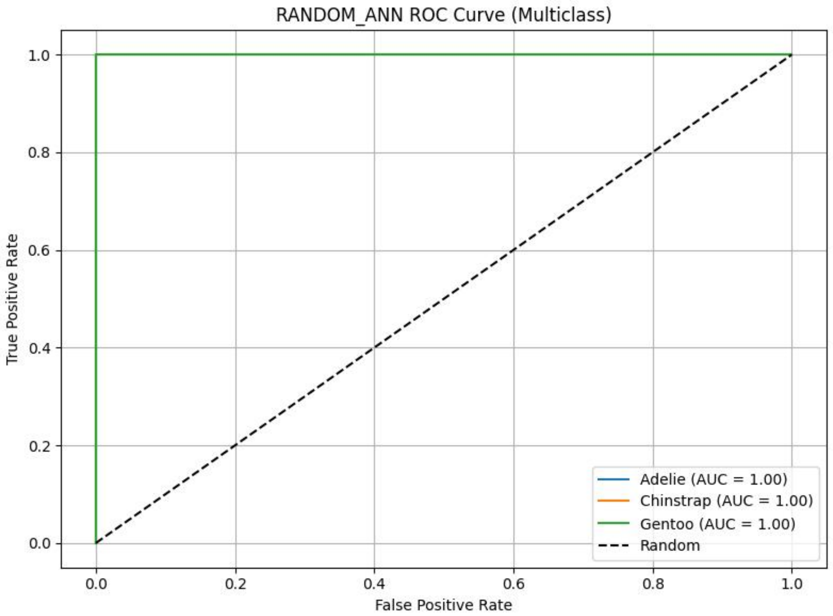
RANDOM_ANN Confusion Matrix:

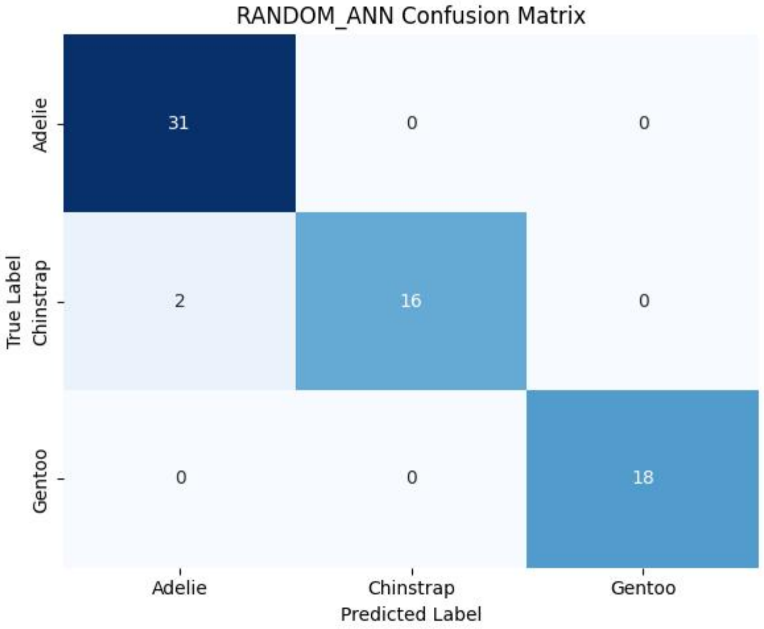
	Adelie	Chinstrap	Gentoo
Adelie	31	0	0
Chinstrap	2	16	0
Gentoo	0	0	18

RANDOM_ANN Average ROC AUC: 1.0000

RANDOM_ANN Evaluation Metrics:

Accuracy: 0.9701
Precision: 0.9798
Recall: 0.9630
Specificity (avg): 0.9815
F1 Score: 0.9700

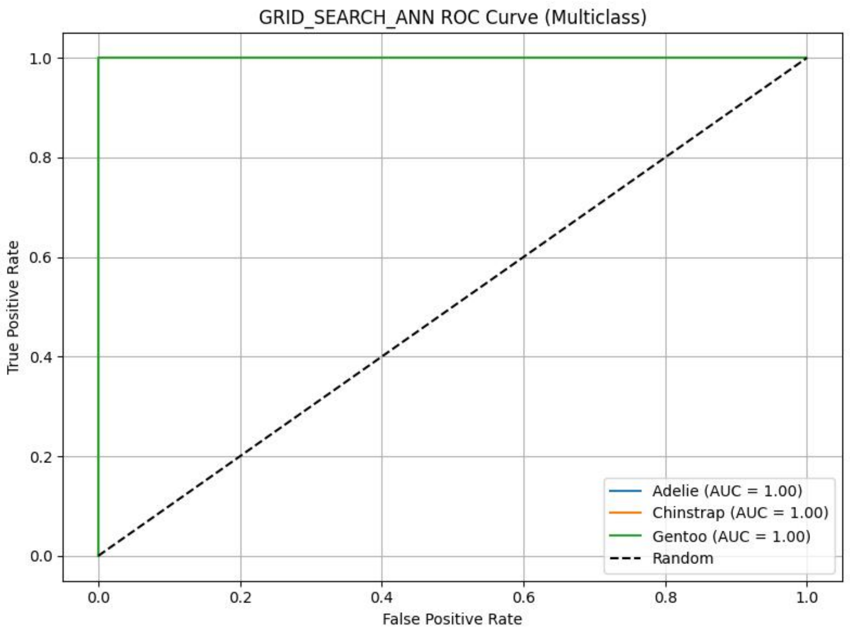


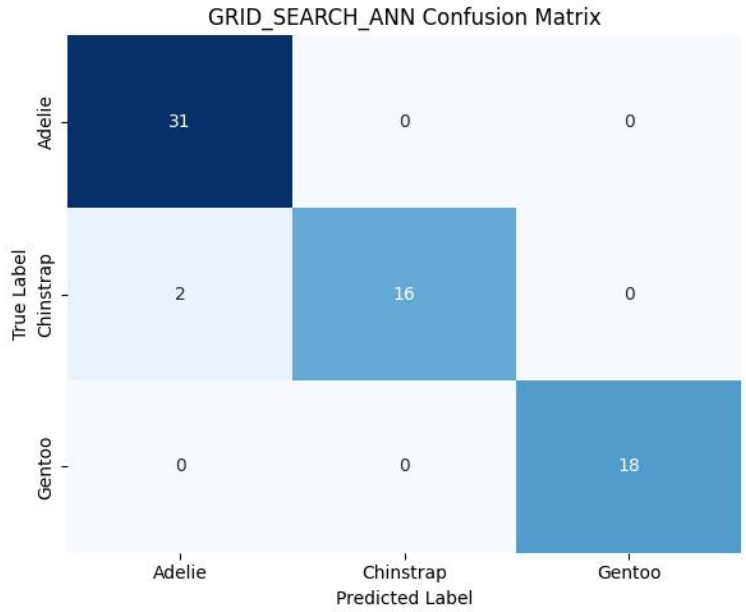


Performance of the model

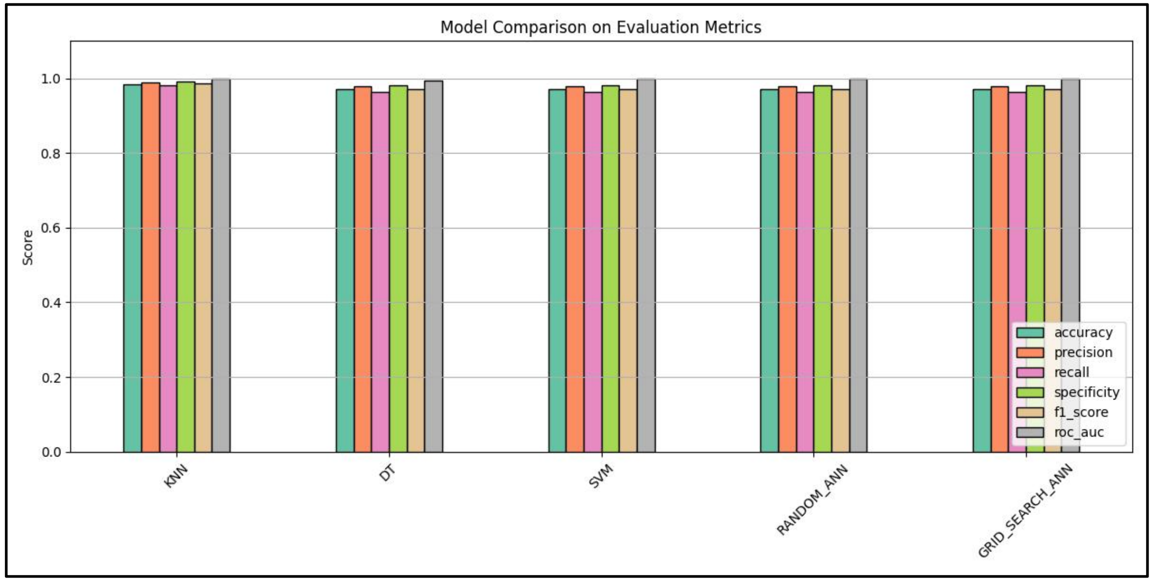
Performance plots for GRID_SEARCH_ANN

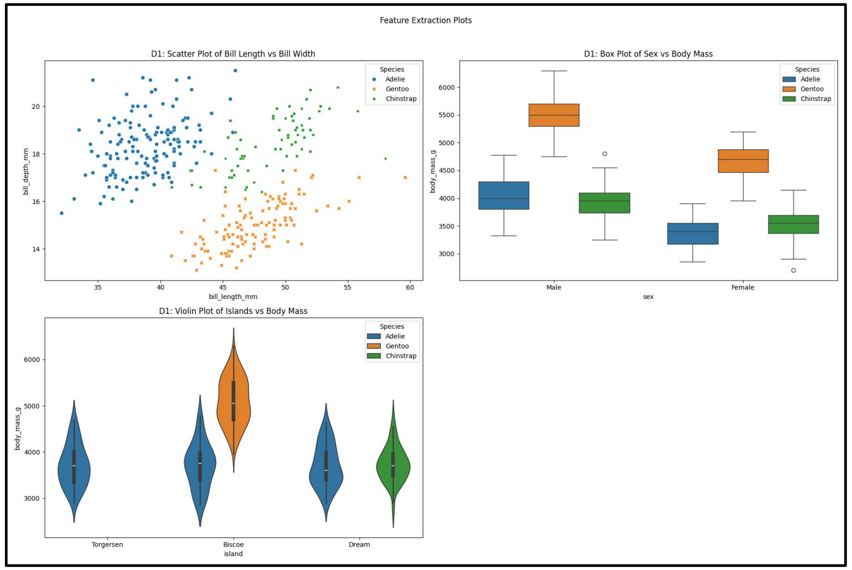
```
GRID_SEARCH_ANN Confusion Matrix:
            Adelie  Chinstrap  Gentoo
Adelie         31          0       0
Chinstrap       2         16       0
Gentoo          0          0      18
GRID_SEARCH_ANN Average ROC AUC: 1.0000

GRID_SEARCH_ANN Evaluation Metrics:
Accuracy:  0.9701
Precision: 0.9798
Recall:    0.9630
Specificity (avg): 0.9815
F1 Score:  0.9700
```





Performance of the model





Discussion:

We found that all models are working very well with this penguins classification task.

We have been very careful to avoid data leakage, and tried severl ways to reduce the accuracy:

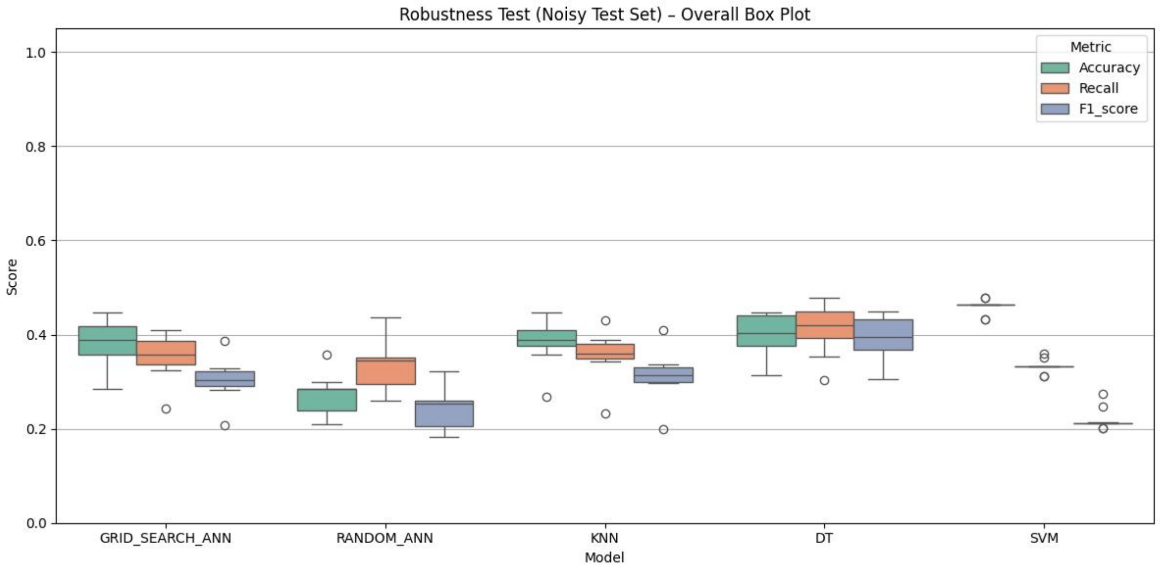
Add noise to the dataset,
limit the nodes and layers of ANN,
drop the island column of the dataset.

Still, the penguins are seperable in some feature, making the task easy to complete.

A	B	C	D	E	F	G
	accuracy	precision	recall	specificity	f1_score	roc_auc
KNN	0.985074627	0.989583333	0.981481481	0.990740741	0.985185185	1
DT	0.970149254	0.97979798	0.962962963	0.981481481	0.96997549	0.994587
SVM	0.970149254	0.97979798	0.962962963	0.981481481	0.96997549	1
RANDOM_ANN	0.970149254	0.97979798	0.962962963	0.981481481	0.96997549	1
GRID_SEARCH_ANN	0.970149254	0.97979798	0.962962963	0.981481481	0.96997549	1

Performance of the model

Robustness test




```python
def noisify(df):  1 usage
    noise_std_dict = {
        'bill_length_mm': 3.0,
        'bill_depth_mm': 1.5,
        'flipper_length_mm': 7.0,
        'body_mass_g': 150.0
    }
```

Discussion:

We conducted 10 trials to test the robustness of the models.

The result decreased significantly because we added strong noise to the test dataset.

Team members & roles / Task completion report

Team members: Drew Hutchinson, Zichuo Wang

MS only tasks were done by Zichuo Wang.

Besides, all the work was accomplished through our discussion, from proposal to the final report writting.

Date	Task Name	Status	Person
3/14/2025	Proposal	Done	Drew Hutchinson, Zichuo Wang
3/31/2025	PA1	Done	Drew Hutchinson, Zichuo Wang
4/14/2025	PA2	Done	Drew Hutchinson, Zichuo Wang
5/8/2025	MS only part	Done	Zichuo Wang
5/8/2025	Coding and fixing	Done	Drew Hutchinson, Zichuo Wang
5/8/2025	Final report writting	Done	Zichuo Wang
5/8/2025	Review and check	Done	Drew Hutchinson