Lazy Predict

In [1]:

```
import seaborn as sns
import pyforest
import warnings
warnings.filterwarnings("ignore")
from sklearn import metrics
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
```

In [2]:

```
iris = sns.load_dataset('iris')
iris.head()
```

Out[2]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [3]:

```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [4]:

```
iris.describe()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

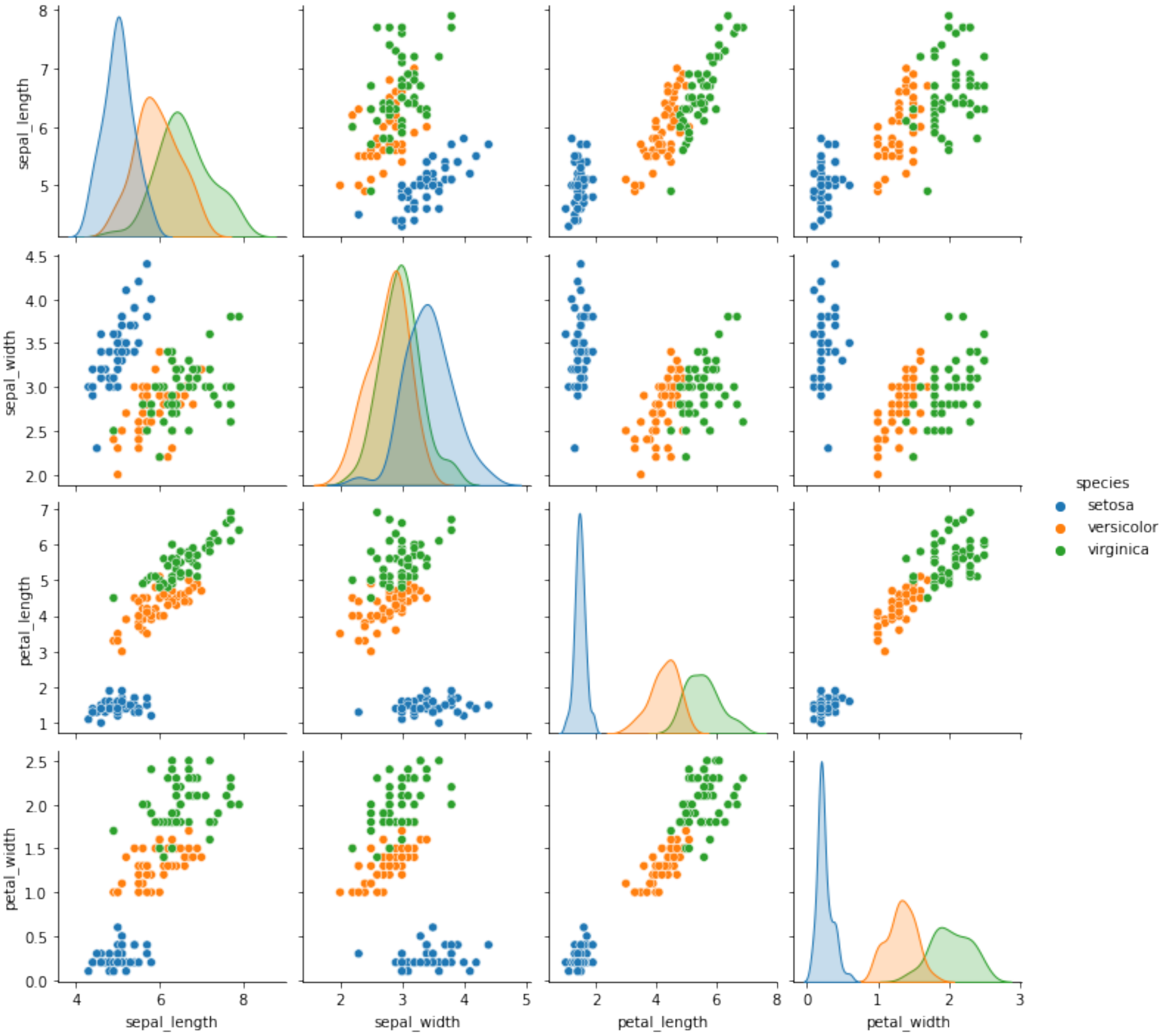
Exploratory Data Analysis

In [5]:

```
sns.pairplot(iris, hue = 'species')
```

Out[5]:

```
<seaborn.axisgrid.PairGrid at 0x7fa972440700>
```



In [6]:

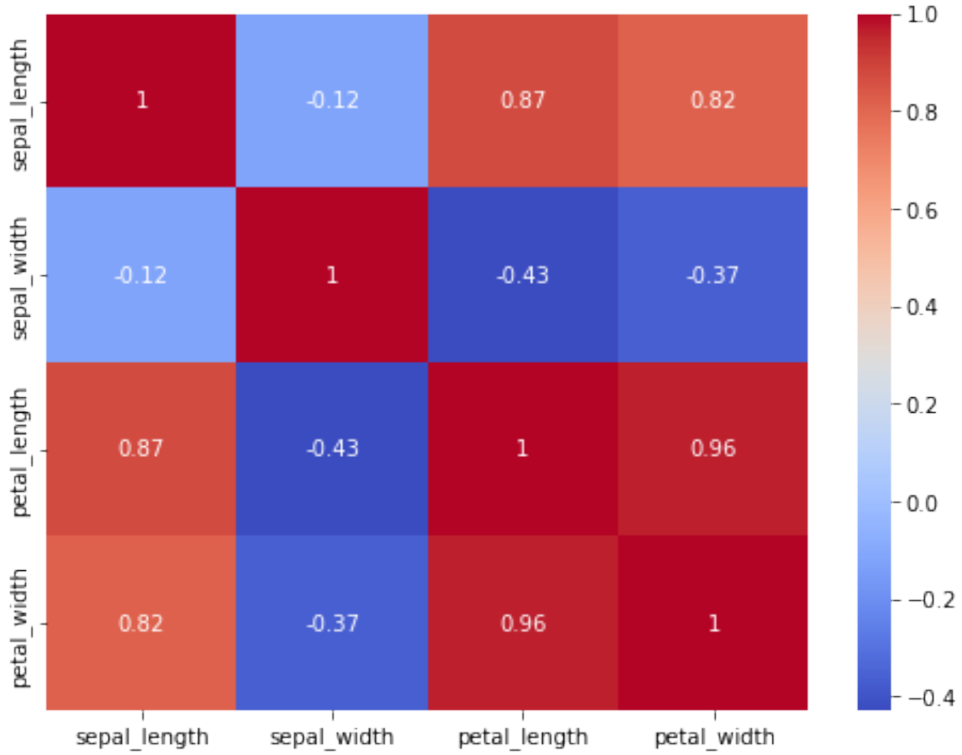
```
iris.columns
```

Out[6]:

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [7]:
```
df = iris[['sepal_length', 'sepal_width', 'petal_length', 'petal_widt
plt.figure(figsize =(8, 6))
sns.heatmap(df.corr(), annot = True, cmap = 'coolwarm')
```

Out[7]:
```
<AxesSubplot:>
```



In [8]:
```
iris['species'].unique()
```

Out[8]:
```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

Lazy Predict

In [9]:
```
# Define X and y

X = iris[['sepal_length', 'sepal_width', 'petal_length', 'petal_width'
y = iris['species']
```

In [10]:
```
# Split the data into Training and Testing data

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test  = train_test_split(X, y, test_size =
```

In [11]:
```
# Modeling with LazyPredict

import lazypredict
from lazypredict.Supervised import LazyClassifier

clf = LazyClassifier()
```

In [12]:
```
# Fit the models and do predictions

models, predictions = clf.fit(X_train, X_test, y_train, y_test)
```

```
100%|████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████| 29/29 [0
0:00<00:00, 36.32it/s]
```

In [13]:
```
print(models)
```

```
                               Accuracy  Balanced Accuracy ROC AUC  F
1 Score  \
Model
QuadraticDiscriminantAnalysis      1.00               1.00    None
1.00
LinearDiscriminantAnalysis         1.00               1.00    None
1.00
LogisticRegression                 0.98               0.98    None
0.98
SVC                                0.98               0.98    None
0.98
ExtraTreesClassifier               0.98               0.98    None
0.98
NuSVC                              0.98               0.98    None
0.98
ExtraTreeClassifier                0.98               0.97    None
0.98
KNeighborsClassifier               0.98               0.97    None
0.98
LGBMClassifier                     0.96               0.96    None
0.96
SGDClassifier                      0.96               0.96    None
0.96
DecisionTreeClassifier             0.96               0.96    None
0.96
GaussianNB                         0.96               0.96    None
0.96
BaggingClassifier                  0.96               0.96    None
0.96
LabelSpreading                     0.96               0.94    None
0.95
LinearSVC                          0.93               0.93    None
```

```
0.93
LabelPropagation                   0.93               0.93    None
0.93
RandomForestClassifier             0.93               0.93    None
0.93
AdaBoostClassifier                 0.93               0.93    None
0.93
Perceptron                         0.91               0.91    None
0.91
PassiveAggressiveClassifier        0.91               0.90    None
0.91
NearestCentroid                    0.84               0.85    None
0.85
CalibratedClassifierCV             0.82               0.85    None
0.83
BernoulliNB                        0.78               0.82    None
0.77
RidgeClassifier                    0.78               0.81    None
0.78
RidgeClassifierCV                  0.78               0.81    None
0.78
DummyClassifier                    0.27               0.33    None
0.11

                               Time Taken
Model
QuadraticDiscriminantAnalysis        0.01
LinearDiscriminantAnalysis           0.01
LogisticRegression                   0.01
SVC                                  0.01
ExtraTreesClassifier                 0.09
NuSVC                                0.01
ExtraTreeClassifier                  0.01
KNeighborsClassifier                 0.01
LGBMClassifier                       0.19
SGDClassifier                        0.01
DecisionTreeClassifier               0.01
GaussianNB                           0.01
BaggingClassifier                    0.03
LabelSpreading                       0.01
LinearSVC                            0.01
LabelPropagation                     0.01
RandomForestClassifier               0.14
AdaBoostClassifier                   0.11
Perceptron                           0.01
PassiveAggressiveClassifier          0.01
NearestCentroid                      0.01
CalibratedClassifierCV               0.04
BernoulliNB                          0.01
RidgeClassifier                      0.01
```

```
RidgeClassifierCV                    0.01
DummyClassifier                      0.01
```

In [14]:
```
# Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier
```

In [15]:
```
dtree = DecisionTreeClassifier()
```

In [16]:
```
dtree.fit(X_train, y_train)
```

Out[16]:
```
DecisionTreeClassifier()
```

In [17]:
```
pred = dtree.predict(X_test)
```

In [18]:
```
from sklearn.metrics import classification_report, confusion_matrix

print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        13
  versicolor       0.95      0.95      0.95        20
   virginica       0.92      0.92      0.92        12

    accuracy                           0.96        45
   macro avg       0.96      0.96      0.96        45
weighted avg       0.96      0.96      0.96        45
```

In [19]:
```
print(confusion_matrix(y_test, pred))
```

```
[[13  0  0]
 [ 0 19  1]
 [ 0  1 11]]
```

In [20]:
```
# Random Forest Classifier

from sklearn.ensemble import RandomForestClassifier
```

In [21]:
```
rfc = RandomForestClassifier(n_estimators = 100)
```

In [22]:
```




rfc
```

Out[22]:
```
RandomForestClassifier()
```

In [23]:
```
rfc.fit(X_train, y_train)
```

Out[23]:
```
RandomForestClassifier()
```

In [24]:
```
rfc_pred = rfc.predict(X_test)
```

In [25]:
```
print(classification_report(y_test, rfc_pred))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        13
  versicolor       0.95      0.95      0.95        20
   virginica       0.92      0.92      0.92        12

    accuracy                           0.96        45
   macro avg       0.96      0.96      0.96        45
weighted avg       0.96      0.96      0.96        45

```

In [26]:
```
print(confusion_matrix(y_test, rfc_pred))
```

```
[[13  0  0]
 [ 0 19  1]
 [ 0  1 11]]
```

In [ ]:
```

```