# Comparison of Random Forest and K-Nearest Neighbor Algorithms for Classifying Varieties of Dry Beans

Carmela Nova N. Edig

## Initializing the code by importing required libraries

In [1]:

```
# Importing the required libraries
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import StratifiedKFold
import numpy as np
from sklearn.metrics import accuracy_score, precision_score
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import scipy.stats as stats
```

## Importing the dataset, which consists of $13611$ points. To proceed with the desired $10-$fold cross-validation, we make sure that the size of each training set is uniform for each fold. Hence, we randomly choose $13610$ datapoints which is divisible by $10$.

In [2]:

```
# Loading dataset from UCI Machine Learning Repository
df = pd.read_excel('Dry_Bean_Dataset.xlsx')
df = df.sample(n=13610)

# Defining the features and target variables
X = df[['Area', 'Perimeter', 'MajorAxisLength',
        'MinorAxisLength', 'AspectRation', 'Eccentricity',
        'ConvexArea', 'EquivDiameter', 'Extent', 'Solidity',
        'roundness', 'Compactness', 'ShapeFactor1',
        'ShapeFactor2', 'ShapeFactor3', 'ShapeFactor4']]
y = df['Class']
```

In [3]:

```
# Getting first 10 rows of column Class from df
df_first_10 = df[['Class']].head(10)

# Printing df_first_10
print(df_first_10)
```

```
          Class
166       SEKER
1233      SEKER
10025      SIRA
4321       CALI
12341  DERMASON
8624       SIRA
2310   BARBUNYA
7338      HOROZ
5703      HOROZ
8108       SIRA
```

## Graphing the distribution of data points per class

It shows that there is an uninform ratio of size per class. Hence, stratified K-fold will be used in training to preserve the distribution.

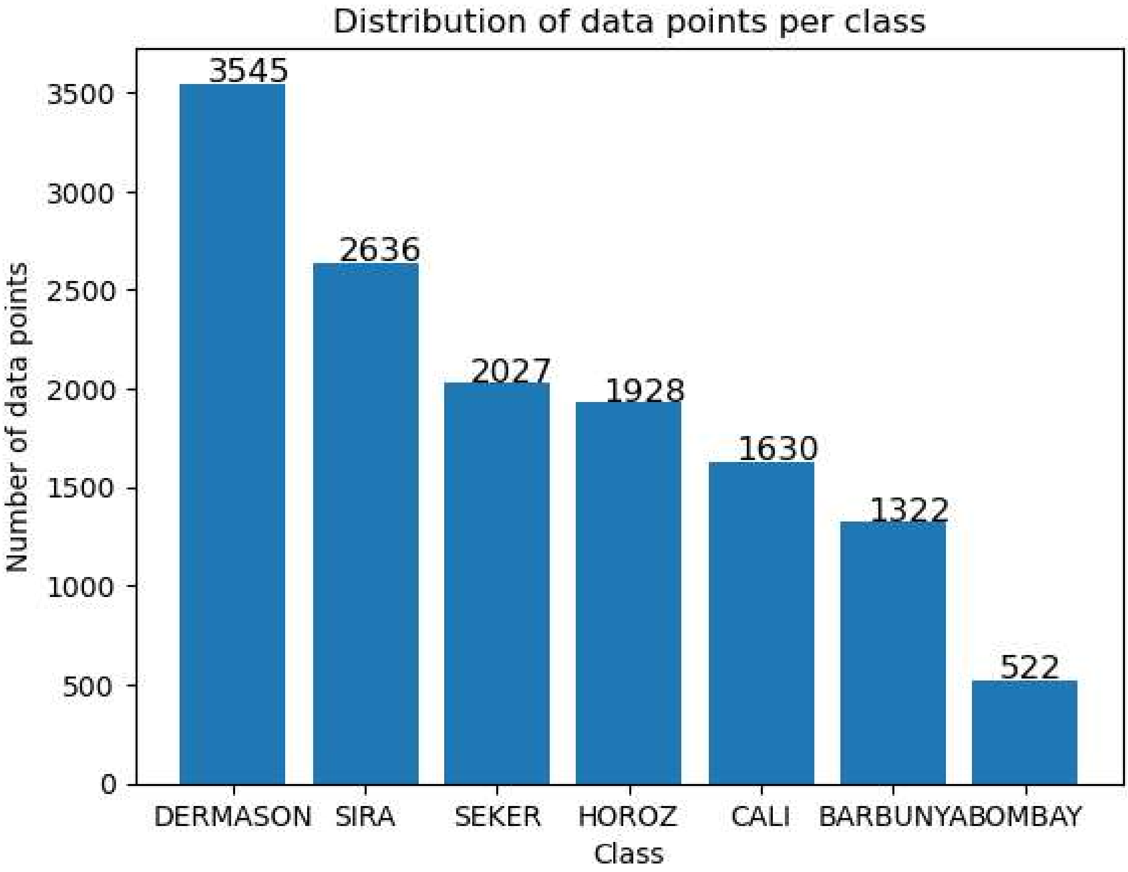
In [4]:

```
# Counting the number of data points per class
class_counts = df['Class'].value_counts()

# Plotting the distribution of data points per class
plt.bar(class_counts.index, class_counts.values)
plt.xlabel('Class')
plt.ylabel('Number of data points')
plt.title('Distribution of data points per class')

# Adding the number of data points per class as a label on top of each bar
for i in range(len(class_counts)):
    plt.text(x = i-0.2 , y = class_counts.values[i]+10, s = class_counts.values[

plt.show()
```



## Training with Random Classifier and $k$-Nearest Neighbors on the same training set generated by each fold, then appending the resulting accuracy and precision metric scores.

In [5]:

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

# Defining the number of folds
n_folds = 10

# Defining the StratifiedKFold object
skf = StratifiedKFold(n_splits=n_folds)

# Creating lists to store the results
rfc_accuracies = []
rfc_precisions = []
knn_accuracies = []
knn_precisions = []
acc_differences = []
pre_differences = []

# Looping through the folds
for i, (train_index, test_index) in enumerate(skf.split(X,y)):
    # Getting the train and test sets for the current fold
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    # Training the Random Forest Classifier
    rfc = RandomForestClassifier()
    rfc.fit(X_train, y_train)
    rfc_accuracy = accuracy_score(y_test, rfc.predict(X_test))
    rfc_precision = precision_score(y_test, rfc.predict(X_test),average='weighte
    rfc_accuracies.append(rfc_accuracy)
    rfc_precisions.append(rfc_precision)

    # Training the K-Nearest Neighbors Classifier
    knn = KNeighborsClassifier()
    knn.fit(X_train, y_train)
    knn_accuracy = accuracy_score(y_test, knn.predict(X_test))
    knn_precision = precision_score(y_test, knn.predict(X_test),average='weighte
    knn_accuracies.append(knn_accuracy)
    knn_precisions.append(knn_precision)

    # Appending the results to the dataframe
    acc_differences.append(rfc_accuracy - knn_accuracy)
    pre_differences.append(rfc_precision - knn_precision)
```

## Printing the results from the previous code

In [6]:

```
# Creating a dataframe from the arrays
results_df = pd.DataFrame({
    'iteration': range(1, n_folds+1),
    'RFC_accuracy': rfc_accuracies,
    'RFC_precision': rfc_precisions,
    'KNN_accuracy': knn_accuracies,
    'KNN_precision': knn_precisions,
    'acc_difference': acc_differences,
    'pre_difference': pre_differences
})

# Printing the table
print(results_df.to_string())

# Averaging accuracy and precision for Random Forest Classifier
rfc_acc_avg = np.mean(rfc_accuracies)
rfc_pre_avg = np.mean(rfc_precisions)

# Averaging accuracy and precision for K-Nearest Neighbors Classifier
knn_acc_avg = np.mean(knn_accuracies)
knn_pre_avg = np.mean(knn_precisions)

# Averaging accuracy difference and precision difference
acc_diff_avg = np.mean(acc_differences)
pre_diff_avg = np.mean(pre_differences)

# Printing results
print("Average accuracy of RFC: %0.4f" % rfc_acc_avg)
print("Average accuracy of KNN: %0.4f" % knn_acc_avg)
print("Average precision of RFC: %0.4f" % rfc_pre_avg)
print("Average precision of KNN: %0.4f" % knn_pre_avg)
print("Average accuracy difference: %0.4f" % acc_diff_avg)
print("Average precision difference:  %0.4f" % acc_diff_avg)
```

```
   iteration  RFC_accuracy  RFC_precision  KNN_accuracy  KNN_precision  acc_dif
ference  pre_difference
0          1      0.929464       0.930393      0.736223       0.736776
0.193240        0.193617
1          2      0.925790       0.926378      0.742836       0.742284
0.182954        0.184094
2          3      0.913299       0.912892      0.733284       0.733164
0.180015        0.179729
3          4      0.933137       0.933412      0.731815       0.733264
0.201323        0.200148
4          5      0.924320       0.924334      0.730345       0.728484
0.193975        0.195849
5          6      0.927259       0.927318      0.727406       0.730941
0.199853        0.196377
6          7      0.925790       0.926308      0.732550       0.736569
0.193240        0.189739
7          8      0.923586       0.923500      0.726672       0.725876
0.196914        0.197624
8          9      0.922116       0.922266      0.736958       0.735665
0.185158        0.186602
9         10      0.919177       0.919217      0.739897       0.742194
0.179280        0.177022
Average accuracy of RFC: 0.9244
Average accuracy of KNN: 0.7338
Average precision of RFC: 0.9246
Average precision of KNN: 0.7345
Average accuracy difference: 0.1906
Average precision difference:  0.1906
```

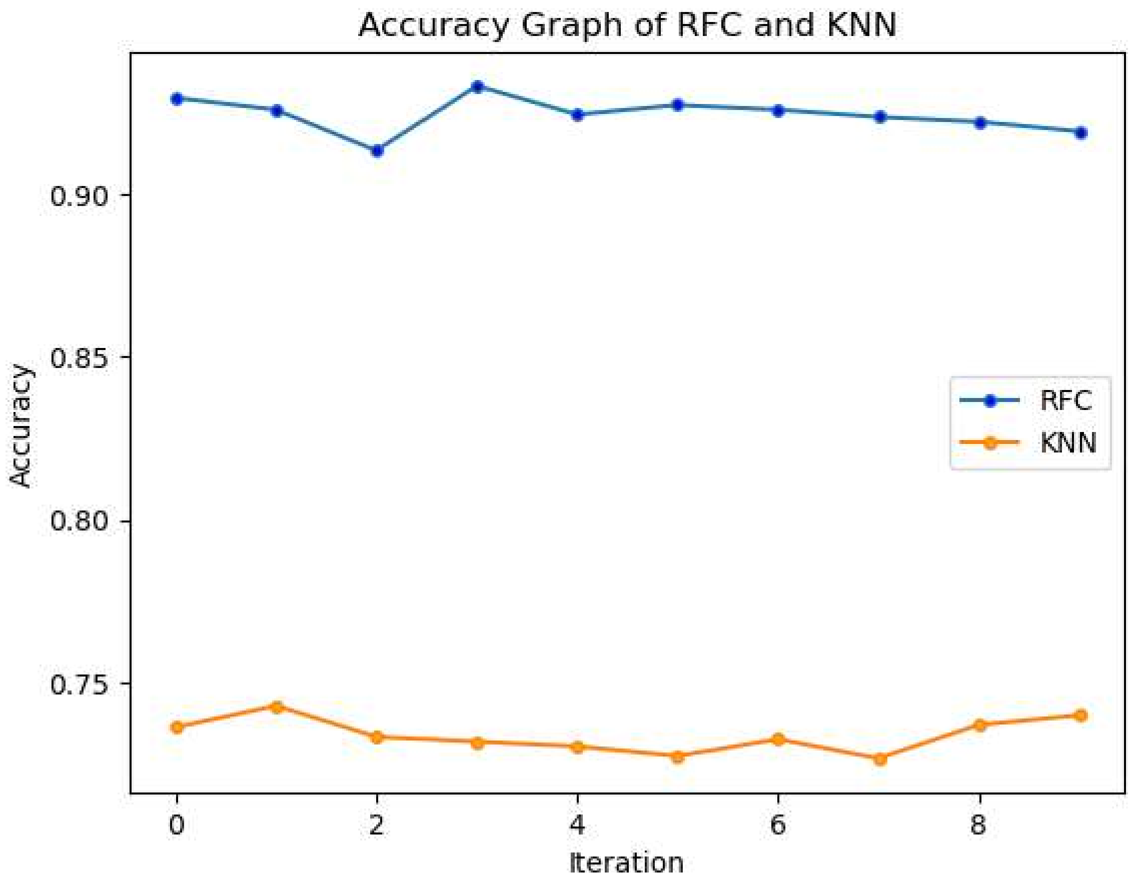
## Graphing results on a line graph

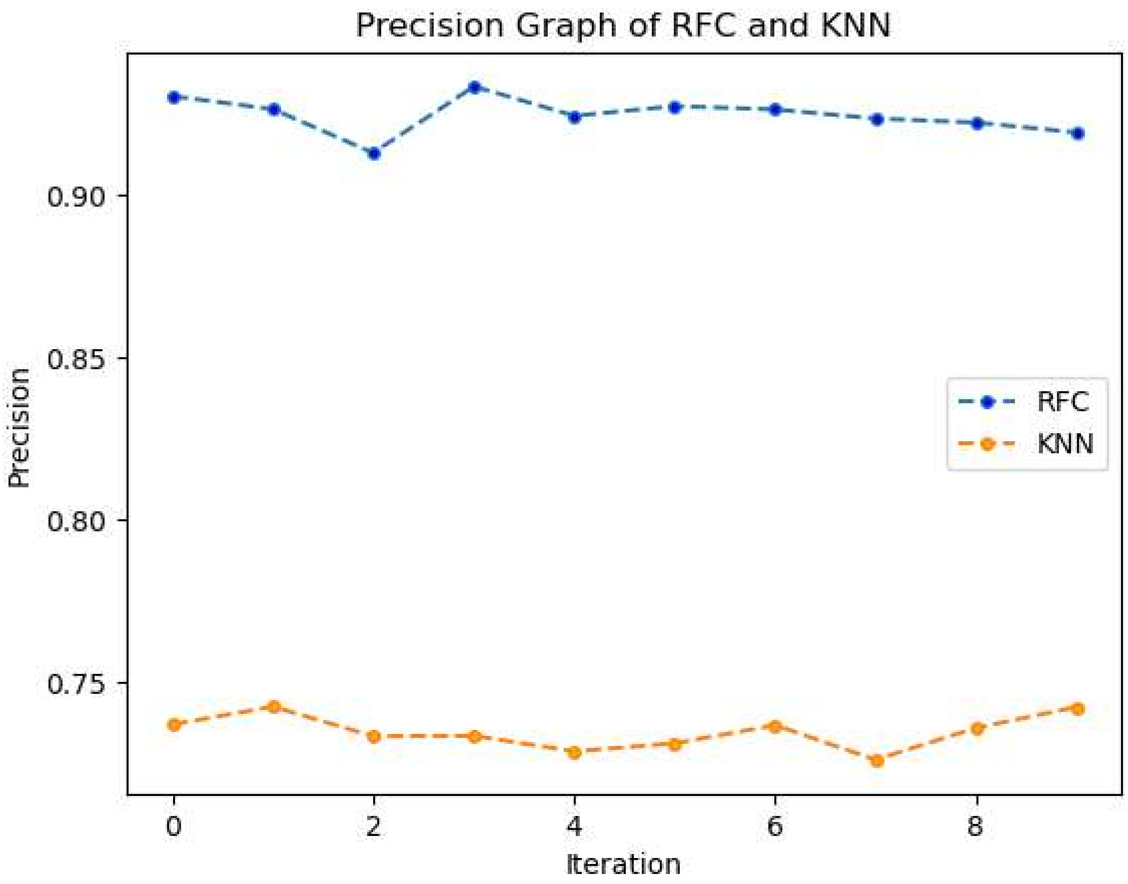
In [7]:

```
# Creating data
x = list(range(0,10))
y = rfc_accuracies
z = knn_accuracies
a = rfc_precisions
b = knn_precisions


plt.xlabel('Iteration')
plt.ylabel('Accuracy')
plt.title('Accuracy Graph of RFC and KNN')
plt.plot(x, y, label = "RFC", marker='o', markerfacecolor='blue', markersize=4)
plt.plot(x, z, label = "KNN", marker='o', markerfacecolor='orange', markersize=4
plt.legend()
plt.show()

plt.xlabel('Iteration')
plt.ylabel('Precision')
plt.title('Precision Graph of RFC and KNN')
plt.plot(x, a, label = "RFC", marker='o', linestyle='dashed', markerfacecolor='b
plt.plot(x, b, label = "KNN", marker='o', linestyle='dashed', markerfacecolor='o
plt.legend()
plt.show()
```





# Statistical Test

## Performing paired t-test

On the accuracy of RFC and KNN:

```
In [8]:  stats.ttest_rel(rfc_accuracies, knn_accuracies)

Out[8]:  Ttest_relResult(statistic=74.18497174438608, pvalue=7.432318144508117e-14)
```

On the precision of RFC and KNN:

```
In [9]:  stats.ttest_rel(rfc_precisions, knn_precisions)

Out[9]:  Ttest_relResult(statistic=75.5693849986105, pvalue=6.294482555962406e-14)
```