CMPT 423/820

Assignment 4 Question 1

- Model Solution and Grading Scheme
- 12 marks

Prologue: Importing Modules

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

from sklearn.naive_bayes import GaussianNB
from sklearn import model_selection

import warnings
warnings.filterwarnings("ignore", category=UserWarning)
```

Prologue: Reading the Data

One of the data files had a different format. This was not a deliberate choice.

In [29]:
```
# a list of names for the columns in the data

filenames = ['iris', 'iris01', 'iris05', 'iris10', 'iris20']
cnames=['SepalLengthCm', 'SepalWidthCm',
        'PetalLengthCm', 'PetalWidthCm', 'Species']
# set up a dictionary to hold all the data
# they are all pretty small

dataframes = {}

# the first file format is different.  Yuck.
dataframes['iris'] = pd.read_csv('A4Q1/iris.csv', header=None,
                         names=cnames, index_col=False)

for fn in filenames[1:]:
    dataframes[fn] = pd.read_csv('A4Q1/'+fn+'.csv',
                             index_col=0)
```

Dealing with missing data

The four methods to fill in missing data are implemented as functions. For the most part, doing this part of the question was a matter of finding the right functions and methods in the Pandas API.

In [28]:
```
def remove_missing(adf):
    """ Remove any line from the dataframe that has a blank entry NA
        Returns a new dataframe
    """
    return adf.dropna()


def fill_randomly(adf):
    """ Any blank entry NA in a dataframe is filled randomly
        Returns a new dataframe
    """
    M = len(adf.index)
    N = len(adf.columns)
    ran = pd.DataFrame(np.random.rand(M,N), columns=adf.columns, index
=adf.index)

    # make a copy, because update modifies the give dataframe
    new_df = adf.copy()
    new_df.update(ran*adf.mean(), overwrite=False)
    return new_df


def fill_column_mean(adf):
    """ Any blank entry NA in a dataframe is filled with the column me
an
        Returns a new dataframe
    """
    new_df = adf.fillna(adf.mean())
    return new_df


def fill_column_class_mean(adf):
    """ Any blank entry NA in a dataframe is filled with the column me
an for that class
        Returns a new dataframe
    """

    setosa = adf[adf['Species'] == 'setosa']
    setosa = setosa.fillna(setosa.mean())

    virginica = adf[adf['Species'] == 'virginica']
    virginica = virginica.fillna(virginica.mean())

    versicolor = adf[adf['Species'] == 'versicolor']
    versicolor = versicolor.fillna(versicolor.mean())

    new_df = pd.concat([setosa,versicolor,virginica], axis=0)
    return new_df
```

Running the techniques, and checking accuracy

With these functions, we simply run all of the data files through them.

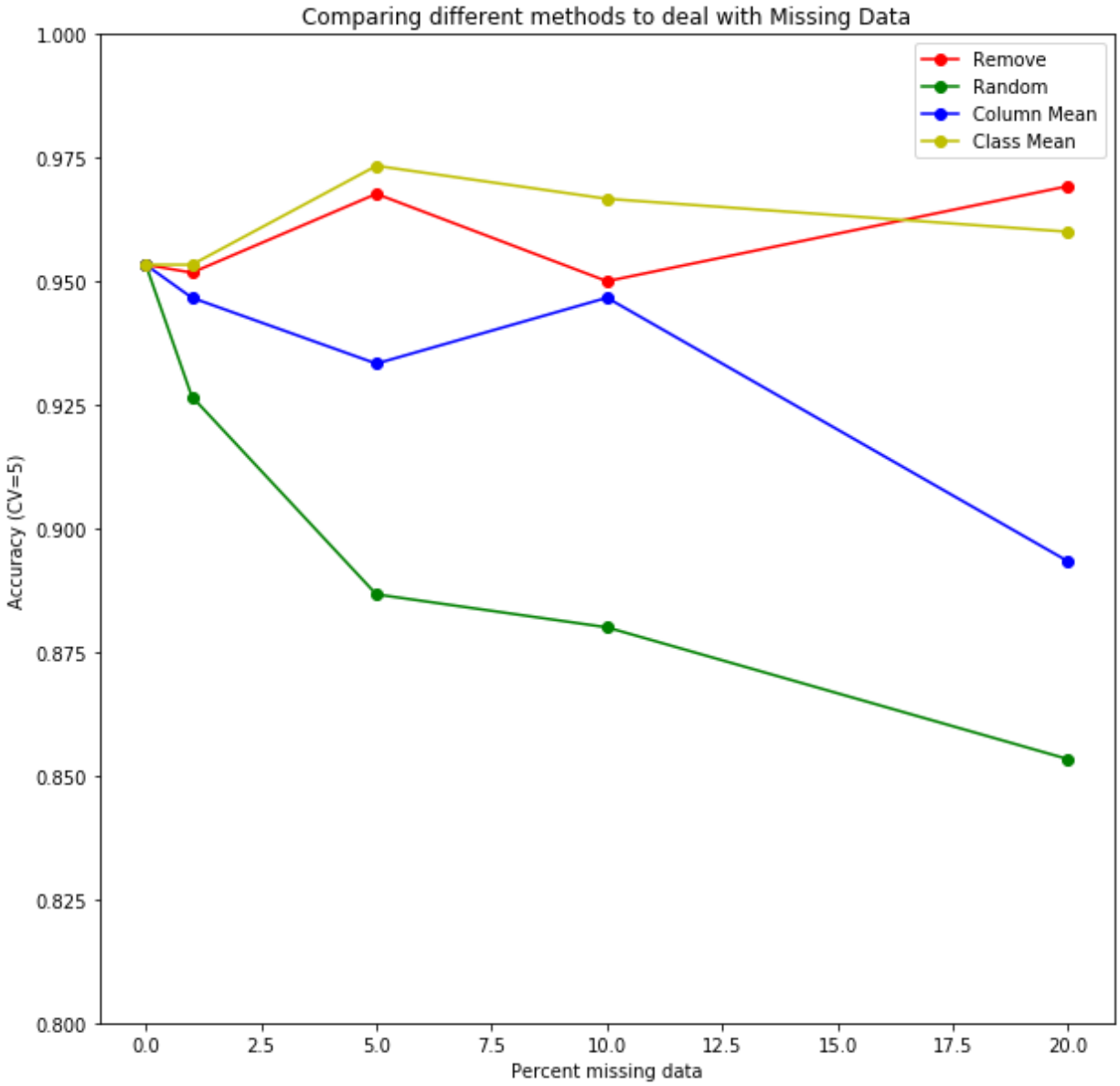
In [42]:
```
def fitNB_and_report(adf):
    """ Normal fitting of Naive Bayes with 5-fold cross-validation.
        Returns the mean and standard deviation of accuracy.
    """
    # Separate the features from the class
    array = adf.values
    X = array[:,0:4]
    Y = array[:,4]

    # cross validation to evaluate the performance using accuracy
    kfold = 5
    cv_results = model_selection.cross_val_score(GaussianNB(), X, Y, c
v=kfold, scoring='accuracy')
    return cv_results.mean(), cv_results.std()

removed           =  [fitNB_and_report( remove_missing(dataframes[fn])
)         for fn in dataframes]
mean_filled       =  [fitNB_and_report( fill_column_mean(dataframes[fn
]))       for fn in dataframes]
class_mean_filled =  [fitNB_and_report( fill_column_class_mean(datafra
mes[fn])) for fn in dataframes]
randomly          =  [fitNB_and_report( fill_randomly(dataframes[fn]))
for fn in dataframes]
```

In [45]:
```
percent_missing = [0, 1, 5, 10, 20]

plt.figure(figsize=(10,10))
plt.plot(percent_missing, [m for m,s in removed],           'ro-')
plt.plot(percent_missing, [m for m,s in randomly],          'go-')
plt.plot(percent_missing, [m for m,s in mean_filled],       'bo-' )
plt.plot(percent_missing, [m for m,s in class_mean_filled], 'yo-')
plt.ylim(0.8, 1.0)
plt.title("Comparing different methods to deal with Missing Data")
plt.ylabel('Accuracy (CV=5)')
plt.xlabel('Percent missing data')
plt.legend(['Remove', 'Random', 'Column Mean', 'Class Mean'])
plt.show()
```

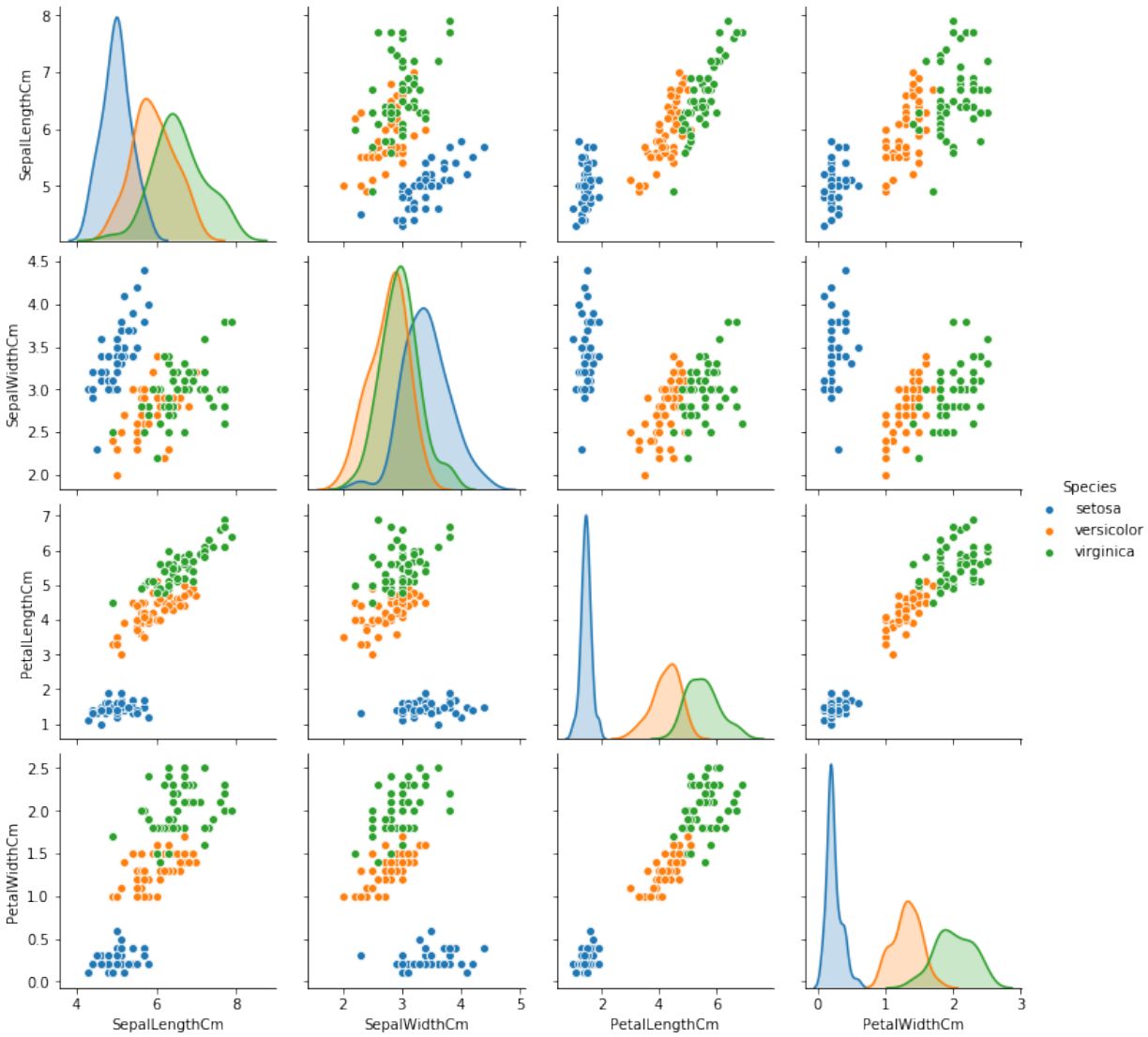


Commentary

We can understand this result by considering what the different techniques do to data points. If you think back to A2, you might remember the Pair-Plot figure showing each pair of features plotted in 2D. The classes appeared as clouds of points. Sometimes the clouds were separated, and sometimes they were over-lapping, to various degrees.

Here's a reminder of those plots. Notice the clouds of data. Imagine slidine any particular point one a line parallel to one of the axes. We'll need that idea below.

In [44]:
```
sns.pairplot(dataframes['iris'], hue="Species", diag_kind='kde')
plt.show()
```



Let's consider each technique individually.

1. **Remove**: This technique simply reduced the number of points. Because the process I used to put holes into the data was essentially random, the missing data was roughly equally split between the three classes. There were fewer data points in each cloud of points. If you take enough points away, you begin to take away spome of the points that resulted in over-lap. Eventually, classes will be easier to classify. But this result is over-fitting: the nice neat classes that result from just a few data points, are sure to make mistakes on the points we know exist in the regions of over-lap. This could be shown using a validation set.
2. **Replace with random.** When we fill a hole with a random data point, we are essentially randomly sliding the point somewhere along the missing column. It's like taking the original data, and sliding the point somewhere parallel to one of the axes in the Pair-Plot. The resulting plots will not form nice neat clouds, and a classifier will have a hard time classifying.
3. **Replace with column mean.** When we fill a hole with the column mean, we are essentially sliding the point to the middle of one of the axes in the Pair-Plot. IT's better than random, but it can result in a central cluster of points that are not the same class.
4. **Replace with column mean for the given class.** When we fill a hole with the class mean, we are essentially sliding the point to the middle of one of the clusters in the Pair-Plot. This is a lot better than the middle of the axis, and it's a lot better than a random place on the axis.

I think it's not surprising that "Random" and "Column Mean" were not as good as "Class Mean." However, it might be surprising that throwing away data results in good classifier accuracy. Being able to see this as a form of over-fitting is key to understanding why we want to avoid it.

Grading -- 12 marks

- 8 marks. Your plot showed the data for the 4 techniques and the 5 data files.
  - It doesn't have to be a single plot. It could be several.
  - The plots are consistent with the solution here.
- 4 marks. You discussed and interpreted the results.
  - You noted that random was bad, and class mean was good.
  - You explained why removing data worked well.
  - There was no need to reproduce the pair plots in the explanation.