Lets import the libraries

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

Lets import the data

In [2]:
```
human_data = pd.read_table(r'C:\Users\SHREE\Downloads\Python CODES\DNA Sequencing using NI
human_data.head()
```

Out[2]:

	sequence	class
0	ATGCCCCAACTAAATACTACCGTATGGCCCACCATAATTACCCCCA...	4
1	ATGAACGAAAATCTGTTCGCTTCATTCATTGCCCCCACAATCCTAG...	4
2	ATGTGTGGCATTTGGGCGCTGTTTGGCAGTGATGATTGCCTTTCTG...	3
3	ATGTGTGGCATTTGGGCGCTGTTTGGCAGTGATGATTGCCTTTCTG...	3
4	ATGCAACAGCATTTTGAATTTGAATACCAGACCAAAGTGGATGGTG...	3

We have some data for human DNA sequence coding regions and a class label.

Let's define a function to collect all possible overlapping k-mers of a specified length from any sequence string.

In [3]:
```
# function to convert sequence strings into k-mer words, default size = 6 (hexamer words)
def getKmers(sequence, size=6):
    return [sequence[x:x+size].lower() for x in range(len(sequence) - size + 1)]
```

Now we can convert our training data sequences into short overlapping k-mers of legth 6. Lets do that for each species of data we have using our getKmers function.

In [4]:
```
human_data['words'] = human_data.apply(lambda x: getKmers(x['sequence']), axis=1)
human_data = human_data.drop('sequence', axis=1)
```

Now, our coding sequence data is changed to lowercase, split up into all possible k-mer words of length 6 and ready for the next step. Let's take a look.

In [5]:
```
human_data.head()
```

Out[5]:

	class	words
0	4	[atgccc, tgcccc, gcccca, ccccaa, cccaac, ccaac...
1	4	[atgaac, tgaacg, gaacga, aacgaa, acgaaa, cgaaa...
2	3	[atgtgt, tgtgtg, gtgtgg, tgtggc, gtggca, tggca...
3	3	[atgtgt, tgtgtg, gtgtgg, tgtggc, gtggca, tggca...
4	3	[atgcaa, tgcaac, gcaaca, caacag, aacagc, acagc...

Since we are going to use scikit-learn natural language processing tools to do the k-mer counting, we need to now convert the lists of k-mers for each gene into string sentences of words that the count vectorizer can use.

We can also make a y variable to hold the class labels. Let's do that now.

In [6]:

```python
human_texts = list(human_data['words'])
for item in range(len(human_texts)):
    human_texts[item] = ' '.join(human_texts[item])
y_data = human_data.iloc[:, 0].values
print(human_texts[2])
```

```
atgtgt tgtgtg gtgtgg tgtggc gtggca tggcat ggcatt gcattt catttg atttgg tttggg ttgggc tgggcg
gggcgc ggcgct gcgctg cgctgt gctgtt ctgttt tgtttg gtttgg tttggc ttggca tggcag ggcagt gcagtg
cagtga agtgat gtgatg tgatga gatgat atgatt tgattg gattgc attgcc ttgcct tgcctt gccttt cctttc
ctttct tttctg ttctgt tctgtt ctgttc tgttca gttcag ttcagt tcagtg cagtgt agtgtc gtgtct tgtctg
gtctga tctgag ctgagt tgagtg gagtgc agtgct gtgcta tgctat gctatg ctatga tatgaa atgaag tgaaga
gaagat aagatt agattg gattgc attgca ttgcac tgcaca gcacac cacaca acacag cacaga acagag cagagg
agaggt gaggtc aggtcc ggtcca gtccag tccaga ccagat cagatg agatgc gatgca atgcat tgcatt gcattc
cattcc attccg ttccgt tccgtt ccgttt cgtttt gttttg ttttga tttgag ttgaga tgagaa gagaat agaatg
gaatgt aatgtc atgtca tgtcaa gtcaat tcaatg caatgg aatgga atggat tggata ggatac gataca atacac
tacacc acacca caccaa accaac ccaact caactg aactgc actgct ctgctg tgctgc gctgct ctgctt tgcttt
gctttg ctttgg tttgga ttggat tggatt ggattt gatttc atttca tttcac ttcacc tcaccg caccgg accggt
ccggtt cggttg ggttgg gttggc ttggcg tggcgg ggcggt gcggta cggtag ggtagt gtagtt tagttg agttga
gttgac ttgacc tgaccc gacccg acccgc cccgct ccgctg cgctgt gctgtt ctgttt tgtttg gtttgg tttgga
ttggaa tggaat ggaatg gaatgc aatgca atgcag tgcagc gcagcc cagcca agccaa gccaat ccaatt caattc
aattcg attcga ttcgag tcgagt cgagtg gagtga agtgaa gtgaag tgaaga gaagaa aagaaa agaaat gaaata
aaatat aatatc atatcc tatccg atccgt tccgta ccgtat cgtatt gtattt tatttg atttgt tttgtg ttgtgg
tgtggc gtggct tggctc ggctct gctctg ctctgt tctgtt ctgtta tgttac gttaca ttacaa tacaat acaatg
caatgg aatggt atggtg tggtga ggtgaa gtgaaa tgaaat gaaatc aaatct aatcta atctac tctaca ctacaa
tacaac acaacc caacca aaccat accata ccataa cataag ataaga taagaa aagaag agaaga gaagat aagatg
agatgc gatgca atgcaa tgcaac gcaaca caacag aacagc acagca cagcat agcatt gcattt catttt attttg
ttttga tttgaa ttgaat tgaatt gaattt aatttg atttga tttgaa ttgaat tgaata gaatac aatacc atacca
taccag accaga ccagac cagacc agacca gaccaa accaaa ccaaag caaagt aaagtg aagtgg agtgga gtggat
tggatg ggatgg gatggt atggtg tggtga ggtgag gtgaga tgagat gagata agataa gataat ataatc taatcc
aatcct atcctt tccttc ccttca cttcat ttcatc tcatct catctt atcttt tcttta ctttat tttatg ttatga
tatgac atgaca tgacaa gacaaa acaaag caaagg aaagga aaggag aggagg ggagga gaggaa aggaat ggaatt
gaattg aattga attgag ttgagc tgagca gagcaa agcaaa gcaaac caaaca aaacaa aacaat acaatt caattt
aatttg atttgt tttgta ttgtat tgtatg gtatgt tatgtt atgttg tgttgg gttgga ttggat tggatg ggatgg
gatggt atggtg tggtgt ggtgtg gtgtgt tgtgtt gtgttt tgtttg gtttgc tttgca ttgcat tgcatt gcattt
catttg atttgt tttgtt ttgttt tgtttt gtttta ttttac tttact ttactg tactgg actgga ctggat tggata
ggatac gatact atactg tactgc actgcc ctgcca tgccaa gccaat ccaata caataa aataag ataaga taagaa
aagaaa agaaag gaaagt aaagtg aagtgt agtgtt gtgttc tgttcc gttcct ttcctg tcctgg cctggg ctgggt
tgggta gggtag ggtaga gtagag tagaga agagat gagata agatac gataca atacat tacata acatat catatg
atatgg tatgga atggag tggagt ggagtc gagtca agtcag gtcaga tcagac cagacc agacct gacctt accttt
cctttg ctttgt tttgtt ttgttt tgttta gtttaa tttaaa ttaaag taaagc aaagca aagcaa agcaat gcaatg
caatga aatgac atgaca tgacag gacaga acagaa cagaag agaaga gaagat aagatg agatgg gatgga atggat
tggatt ggattt gatttt attttt tttttg ttttgg tttggc ttggct tggctg ggctgt gctgta ctgtat tgtatg
gtatgt tatgtt atgttc tgttca gttcag ttcaga tcagaa cagaag agaagc gaagct aagcta agctaa gctaaa
ctaaag taaagg aaaggt aaggtc aggtct ggtctt gtcttg tcttgt cttgtt ttgtta tgttac gttaca ttacat
tacatt acattg cattga attgaa ttgaag tgaagc gaagca aagcac agcact gcactc cactcc actccg ctccgc
tccgcg ccgcga cgcgac gcgact cgactc gactcc actccc ctccct tccctt cccttt cctttt cttttt ttttta
ttttaa tttaaa ttaaaa taaaag aaaagt aaagtg aagtgg agtgga gtggag tggagc ggagcc gagcct agcctt
gccttt cctttt cttttc ttttct tttctt ttcttc tcttcc cttcct ttcctg tcctgg cctgga ctggac tggaca
ggacac gacact acacta cactat actatg ctatga tatgaa atgaag tgaagt gaagtt aagttt agtttt gttttg
ttttgg tttgga ttggat tggatt ggattt gattta atttaa tttaaa ttaaag taaagc aaagcc aagcca agccaa
gccaaa ccaaat caaatg aaatgg aatggc atggca tggcaa ggcaaa gcaaag caaagt aaagtt aagttg agttgc
gttgca ttgcat tgcatc gcatcc catccg atccgt tccgtg ccgtgg cgtgga gtggaa tggaaa ggaaat gaaatg
aaatgg aatggt atggtt tggtta ggttaa gttaaa ttaaat taaata aaatat aatatc atatca tatcat atcatc
tcatca catcac atcact tcactg cactgt actgtc ctgtcg tgtcgg gtcggg tcggga cgggat gggatg ggatgt
gatgta atgtac tgtacc gtaccc tacccc acccct cccctg ccctgc cctgca ctgcac tgcacg gcacgc cacgcc
acgccc cgccct gccctc ccctct cctcta ctctat tctatg ctatga tatgac atgaca tgacaa gacaat acaatg
caatgt aatgtg atgtgg tgtgga gtggag tggaga ggagaa gagaaa agaaac gaaact aaactc aactct actctt
ctcttt tctttc ctttcc tttcca ttccag tccagg ccaggt caggtt aggttt ggtttt gttttg ttttga tttgag
ttgaga tgagat gagata agatag gataga atagaa tagaaa agaaac gaaact aaactg aactgt actgtg ctgtga
tgtgaa gtgaag tgaaga gaagaa aagaac agaaca gaacaa aacaac acaacc caacct aacctc acctca cctcag
ctcagg tcagga caggat aggatc ggatcc gatcct atcctt tccttt cctttt cttttt ttttta ttttaa tttaat
ttaata taataa aataat ataatg taatgc aatgct atgctg tgctgt gctgta ctgtaa tgtaaa gtaaag taaaga
```

aaagaa aagaaa agaaac gaaacg aaacgt aacgtt acgttt cgtttg gtttga tttgat ttgatg tgatga gatgac
atgaca tgacag gacaga acagac cagaca agacag gacaga acagaa cagaag agaagg gaagga aaggat aggatt
ggattg gattgg attggc ttggct tggctg ggctgc gctgcc ctgcct tgcctt gccttt cctttt ctttta ttttat
tttatc ttatca tatcag atcagg tcaggg cagggg aggggg gggggc ggggct gggctt ggcttg gcttgg cttgga
ttggac tggact ggactc gactcc actcca ctccag tccagc ccagct cagctt agcttg gcttgg cttggt ttggtt
tggttg ggttgc gttgct ttgctg tgctgc gctgcc ctgcca tgccac gccact ccactc cactct actctg ctctgt
tctgtt ctgttg tgttga gttgaa ttgaag tgaagc gaagca aagcag agcagc gcagct cagctg agctga gctgaa
ctgaaa tgaaag gaaaga aaagaa aagaag agaagc gaagcc aagccc agccca gcccaa cccaag ccaagt caagta
aagtac agtaca gtacag tacagt acagta cagtat agtatc gtatcc tatcct atcctc tcctct cctctc ctctcc
tctcca ctccag tccaga ccagac cagaca agacat gacatt acattt catttg atttgc tttgca ttgcaa tgcaat
gcaatt caattg aattgg attggc ttggca tggcat ggcatg gcatgg catgga atggaa tggaag ggaaga gaagac
aagaca agacag gacagc acagcc cagccc agcccc gccccg ccccga cccgat ccgatt cgattt gattta atttac
tttact ttactg tactgg actggc ctggct tggctg ggctgc gctgct ctgcta tgctag gctaga ctagaa tagaaa
agaaag gaaagg aaaggt aaggtg aggtgg ggtggc gtggca tggcag ggcaga gcagat cagatc agatca gatcat
atcata tcatat catatt atattg tattgg attgga ttggaa tggaag ggaagt gaagtg aagtga agtgaa gtgaac
tgaaca gaacat aacatt acatta cattat attatg ttatga tatgaa atgaag tgaagt gaagtc aagtcc agtcct
gtcctt tccttt cctttt cttttt ttttta ttttaa tttaac ttaact taactc aactct actctg ctctga tctgag
ctgagg tgagga gaggaa aggaag ggaagg gaaggc aaggca aggcat ggcatt gcattc cattca attcag ttcagg
tcaggc caggct aggctc ggctct gctctg ctctgg tctgga ctggat tggatg ggatga gatgaa atgaag tgaagt
gaagtc aagtca agtcat gtcata tcatat catatt atattt tatttt attttc ttttcc tttcct ttcctt tccttg
ccttgg cttgga ttggaa tggaaa ggaaac gaaact aaactt aactta acttat cttatg ttatga tatgac atgaca
tgacat gacatt acatta cattac attaca ttacaa tacaac acaaca caacag aacagt acagtt cagttc agttcg
gttcgt ttcgtg tcgtgc cgtgct gtgctt tgcttc gcttca cttcag ttcagt tcagta cagtag agtagg gtaggt
taggta aggtat ggtatg gtatgt tatgta atgtat tgtatt gtattt tattta atttaa tttaat ttaatt taattt
aatttc atttcc tttcca ttccaa tccaag ccaagt caagta aagtat agtata gtatat tatatt atattc tattcg
attcgg ttcgga tcggaa cggaag ggaaga gaagaa aagaac agaaca gaacac aacaca acacag cacaga acagat
cagata agatag gatagc atagcg tagcgt agcgtg gcgtgg cgtggt gtggtg tggtga ggtgat gtgatc tgatct
gatctt atcttc tcttct cttctc ttctct tctctg ctctgg tctgga ctggag tggaga ggagaa gagaag agaagg
gaagga aaggat aggatc ggatca gatcag atcaga tcagat cagatg agatga gatgaa atgaac tgaact gaactt
aactta acttac cttacg ttacgc tacgca acgcag cgcagg gcaggg cagggt agggtt gggtta ggttac gttaca
ttacat tacata acatat catata atatat tatatt atattt tatttt attttc ttttca tttcac ttcaca tcacaa
cacaag acaagg caaggc aaggct aggctc ggctcc gctcct ctcctt tccttc ccttct cttctc ttctcc tctcct
ctcctg tcctga cctgaa ctgaaa tgaaaa gaaaaa aaaaag aaaagc aaagcc aagccg agccga gccgag ccgagg
cgagga gaggag aggagg ggagga gaggag aggaga ggagag gagagt agagtg gagtga agtgag gtgaga tgagag
gagagg agaggc gaggct aggctt ggcttc gcttct cttctg ttctga tctgag ctgagg tgaggg gaggga agggaa
gggaac ggaact gaactc aactct actcta ctctat tctatt ctattt tatttg atttgt tttgtt ttgttt tgtttg
gtttga tttgat ttgatg tgatgt gatgtt atgttc tgttct gttctc ttctcc tctccg ctccgc tccgcg ccgcgc
cgcgca gcgcag cgcaga gcagat cagatc agatcg gatcga atcgaa tcgaac cgaact gaacta aactac actact
ctactg tactgc actgct ctgctg tgctgc gctgcc ctgccc tgccca gcccat cccatg ccatgg catggt atggtc
tggtct ggtctt gtcttg tcttga cttgaa ttgaac tgaact gaactg aactga actgag ctgaga tgagag gagagt
agagtc gagtcc agtccc gtccca tcccat cccatt ccattt catttc atttct tttcta ttctag tctaga ctagat
tagatc agatca gatcat atcatc tcatcg catcga atcgat tcgatt cgattt gatttt attttc ttttct tttctt
ttcttc tcttcc cttcct ttccta tcctat cctatt ctatta tattac attact ttactt tacttg acttgt cttgtc
ttgtct tgtctc gtctct tctctg ctctgc tctgcc ctgcca tgccac gccacc ccacca caccag accaga ccagaa
cagaaa agaaat gaaatg aaatga aatgag atgaga tgagaa gagaat agaatt gaattc aattcc attcca ttccaa
tccaaa ccaaag caaaga aaagaa aagaat agaatg gaatgg aatggg atggga tgggat gggata ggatag gataga
atagaa tagaaa agaaaa gaaaaa aaaaac aaaaca aaacat aacatc acatct catctc atctcc tctcct ctcctg
tcctga cctgag ctgaga tgagag gagaga agagag gagaga agagac gagacg agacgt gacgtt acgttt cgtttg
gtttga tttgag ttgagg tgagga gaggat aggatt ggattc gattcc attcca ttccaa tccaat ccaatc caatct
aatctg atctga tctgat ctgata tgatac gatacc ataccc taccca acccaa cccaaa ccaaag caaaga aaagag
aagaga agagat gagatt agattc gattct attctc ttctct tctctg ctctgg tctggc ctggcg tggcga ggcgac
gcgacc cgacca gaccaa accaaa ccaaaa caaaag aaaaga aaagaa aagaag agaagc gaagcc aagcct agcctt
gccttc ccttca cttcag ttcagt tcagtg cagtga agtgat gtgatg tgatgg gatgga atggaa tggaat ggaata
gaataa aataac ataact taactt aacttc acttca cttcag ttcagt tcagtt cagtta agttaa gttaag ttaaga
taagaa aagaat agaatt gaattc aattcc attcct ttcctg tcctgg cctggt ctggtt tggttt ggttta gtttaa
tttaag ttaaga taagat aagatt agattt gatttt atttta ttttac tttaca ttacag tacagg acagga caggaa
aggaat ggaata gaatac aatacg atacgt tacgtt acgttg cgttga gttgaa ttgaac tgaaca gaacat aacatc
acatca catcag atcagg tcaggt caggtt aggttg ggttga gttgat ttgatg tgatga gatgat atgatg tgatgc
gatgca atgcaa tgcaat gcaatg caatga aatgat atgatg tgatgg gatggc atggca tggcaa ggcaaa gcaaat
caaatg aaatgc aatgca atgcag tgcagc gcagcc cagccc agccca gcccag cccaga ccagaa cagaaa agaaat
gaaatt aaattt aatttc atttcc tttccc ttccct tccctt cccttc ccttca cttcaa ttcaat tcaata caatac
aatact atactc tactcc actcct ctccta tcctaa cctaaa ctaaaa taaaac aaaacc aaacca aaccaa accaaa
ccaaag caaaga aaagaa aagaag agaagg gaagga aaggat aggata ggatat gatatt atatta tattac attact
ttacta tactac actacc ctaccg taccgt accgtc ccgtca cgtcaa gtcaag tcaagt caagtc aagtct agtctt
gtcttt tctttg ctttga tttgaa ttgaac tgaacg gaacgc aacgcc acgcca cgccat gccatt ccatta cattac
attacc ttaccc taccca acccag cccagg ccaggc caggcc aggccg ggccgg gccggg ccgggc cgggct gggctg

```
ggctga gctgac ctgact tgactg gactgg actggc ctggct tggctg ggctga gctgag ctgagc tgagcc gagcca
agccat gccatt ccatta cattac attact ttactg tactgg actgga ctggat tggatg ggatgc gatgcc atgccc
tgccca gcccaa cccaag ccaagt caagtg aagtgg agtgga gtggat tggatc ggatca gatcaa atcaat tcaatg
caatgc aatgcc atgcca tgccac gccact ccactg cactga actgac ctgacc tgaccc gaccct accctt cccttc
ccttct cttctg ttctgc tctgcc ctgccc tgcccg gcccgc cccgca ccgcac cgcacg gcacgc cacgct acgctg
cgctga gctgac ctgacc tgaccc gaccca acccac cccact ccacta cactac actaca ctacaa tacaag acaagt
caagtc aagtca agtcag gtcagc tcagct cagctg agctgt gctgtc ctgtca tgtcaa gtcaaa tcaaag caaagc
aaagct aagctt agctta gcttag
```

In [7]:

```
y_data
```

Out[7]:

```
array([4, 4, 3, ..., 6, 6, 6], dtype=int64)
```

We will perform the same steps for chimpanzee and dog

Now we will apply the BAG of WORDS using CountVectorizer using NLP

In [8]:

```
# Creating the Bag of Words model using CountVectorizer()
# This is equivalent to k-mer counting
# The n-gram size of 4 was previously determined by testing
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(ngram_range=(4,4))
X = cv.fit_transform(human_texts)
print(X.shape)
```

```
(4380, 232414)
```

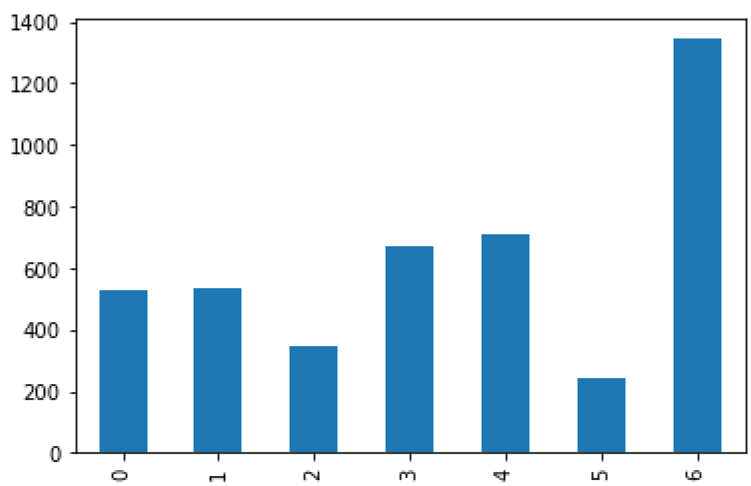
If we have a look at class balance we can see we have relatively balanced data set.

In [9]:

```
human_data['class'].value_counts().sort_index().plot.bar()
```

Out[9]:

```
<AxesSubplot:>
```



Splitting the human data set into the training set and test set

In [10]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y_data,
                                                    test_size = 0.20,
                                                    random_state=42)
print(X_train.shape)
print(X_test.shape)
```

```
(3504, 232414)
(876, 232414)
```

A multinomial naive Bayes classifier will be created. I previously did some parameter tuning and found the ngram size of 4 (reflected in the Countvectorizer() instance) and a model alpha of 0.1 did the best.

In [11]:

```
# The alpha parameter was determined by grid search previously
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB(alpha=0.1)
classifier.fit(X_train, y_train)
```

Out[11]:

```
▾     MultinomialNB
MultinomialNB(alpha=0.1)
```

In [12]:

```
y_pred = classifier.predict(X_test)
```

Let's look at some model performance metrics like the confusion matrix, accuracy, precision, recall and f1 score.

In [13]:

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score
print("Confusion matrix\n")
print(pd.crosstab(pd.Series(y_test, name='Actual'), pd.Series(y_pred, name='Predicted')))
def get_metrics(y_test, y_predicted):
    accuracy = accuracy_score(y_test, y_predicted)
    precision = precision_score(y_test, y_predicted, average='weighted')
    recall = recall_score(y_test, y_predicted, average='weighted')
    f1 = f1_score(y_test, y_predicted, average='weighted')
    return accuracy, precision, recall, f1
accuracy, precision, recall, f1 = get_metrics(y_test, y_pred)
print("accuracy = %.3f \nprecision = %.3f \nrecall = %.3f \nf1 = %.3f" % (accuracy, precis
```

```
Confusion matrix

Predicted   0    1   2    3    4   5    6
Actual
0          99    0   0    0    1   0    2
1           0  104   0    0    0   0    2
2           0    0  78    0    0   0    0
3           0    0   0  124    0   0    1
4           1    0   0    0  143   0    5
5           0    0   0    0    0  51    0
6           1    0   0    1    0   0  263
accuracy = 0.984
precision = 0.984
recall = 0.984
f1 = 0.984
```

We are getting really good results on our unseen data, so it looks like our model did not overfit to the training data.