Predicting Viral Host from Codon Usage Bias

Kathleen Hablutzel

BIO 334/335 Bioinformatics Final Project

This is the same code workflow as the analysis with the full genome.

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

import math
import itertools
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.utils import shuffle
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

# all my helper functions
from helper import *
```

In [2]:
```
# load data
bias_pd = pd.read_csv("datasets/report/cub_biased_genome.csv")
bias_pd.shape
```

Out[2]:
```
(7426, 72)
```

In [3]:
```
bias_pd.head()
```

Out[3]:

	UUU	UUC	UUA	UUG	CUU	CUC	CUA	CUG	AUU
0	0.357377	0.642623	0.035458	0.104243	0.099131	0.213422	0.065184	0.482563	0.277330
1	0.365337	0.634663	0.050157	0.098445	0.096961	0.220268	0.049994	0.484176	0.278849
2	0.371288	0.628712	0.054070	0.102735	0.100069	0.220455	0.050968	0.471703	0.288741
3	0.348407	0.651593	0.044989	0.095981	0.091464	0.225333	0.047891	0.494342	0.273083
4	0.339304	0.660696	0.041498	0.094636	0.088775	0.225838	0.048453	0.500799	0.266361

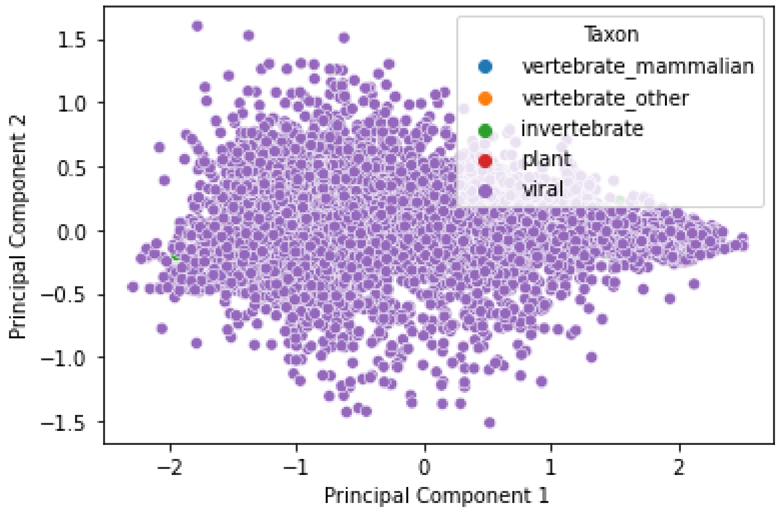
5 rows × 72 columns

In [4]:
```
# store which columns we drop to just get 59 codons
drop_columns = ["AUG", "UGA", "UAA", "UAG", "UGG", "AccessionNum", "SeqLen", "Bi
                "Nc", "BiasedNc", "Species", "Taxon"]
```

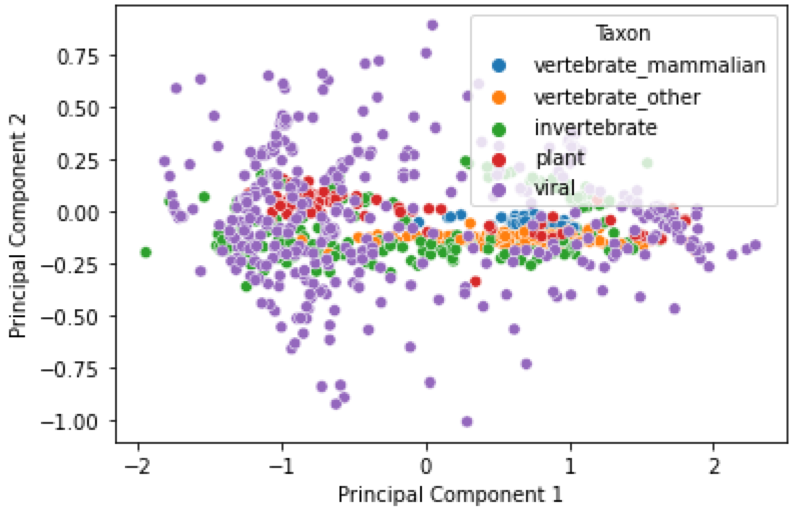
Visualizations

In [5]:
```
# summarize entire dataset in 2d
bias_2d = reduce_dim(bias_pd, dim=2, drop_columns=drop_columns, vertebrate=False
```

In [6]:
```
# plot of all CUB in the dataset - overwhelmed by the number of viruses vs. othe
sns.scatterplot(data=bias_2d, x='Dim1', y='Dim2', hue='Taxon')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2');
```



In [7]:
```
# plot with fewer viruses - still hard to see any meaningful differences
sns.scatterplot(data=bias_2d.head(1000), x='Dim1', y='Dim2', hue='Taxon')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2');
```



Identifying virus host from CUB

Pull in extra viral host data from: https://www.ncbi.nlm.nih.gov/genomes/GenomesGroup.cgi?taxid=10239&cmd=download2 (On https://www.ncbi.nlm.nih.gov/genome/viruses/, it's the

Accession List of all viral genomes)

In [8]:
```
# read in viral data
hosts = pd.read_csv("datasets/viral_hosts.csv", skiprows=[0])
hosts = hosts.drop(columns=["Neighbor", "Segment name", "Representative", "Selec

# remove duplicate entries for each virus
hosts = hosts.drop_duplicates(ignore_index=True).dropna()
print(hosts.shape)
hosts.head(10)
```

```
(16095, 2)
```

Out[8]:

	Host	Taxonomy name
0	human,vertebrates	Cowpox virus
1	human,vertebrates	Monkeypox virus
2	human,vertebrates	Monkeypox virus Zaire-96-I-16
3	human,vertebrates	Vaccinia virus
4	human,vertebrates	Vaccinia virus Copenhagen
5	human,vertebrates	Vaccinia virus Ankara
6	human,vertebrates	Vaccinia virus Tian Tan
7	human,vertebrates	Rabbitpox virus
8	human,vertebrates	Horsepox virus
9	human,vertebrates	Buffalopox virus

In [9]:
```
# follow CUB dataset naming convention
hosts['Species'] = hosts['Taxonomy name'].apply(lambda x: x.replace(" ", "_"))

# column for whether human is host or not
hosts['Human_Host'] = hosts['Host'].apply(lambda x: "human" in x)
hosts.head()
```

Out[9]:

	Host	Taxonomy name	Species	Human_Host
0	human,vertebrates	Cowpox virus	Cowpox_virus	True
1	human,vertebrates	Monkeypox virus	Monkeypox_virus	True
2	human,vertebrates	Monkeypox virus Zaire-96-I-16	Monkeypox_virus_Zaire-96-I-16	True
3	human,vertebrates	Vaccinia virus	Vaccinia_virus	True
4	human,vertebrates	Vaccinia virus Copenhagen	Vaccinia_virus_Copenhagen	True

In [10]:
```
# join in host info where matches any viruses in the CUB dataset
# inner join -> only keep species w/ info in both datasets
virus_cub = pd.merge(bias_pd, hosts, on="Species")
virus_cub.shape
```

Out[10]:
```
(3037, 75)
```

In [11]:

```
# 154 viruses w/ human host
human_sum = np.sum(virus_cub["Human_Host"])
human_sum
```

Out[11]:

```
154
```

In [12]:

```
# keep equal numbers human and non-human viruses
# (otherwise non-human viruses overwhelm the dataset)
np.random.seed(88)
virus_human = virus_cub[virus_cub["Human_Host"]]
virus_non_human = virus_cub[~virus_cub["Human_Host"]]
virus_non_human = shuffle(virus_non_human).head(human_sum)  # keep random viruse

virus_cub_short = pd.concat([virus_human, virus_non_human])
virus_cub_short.shape
```

Out[12]:

```
(308, 75)
```

In [13]:

```
viral_drop_columns = drop_columns + ["Host", "Taxonomy name", "Human_Host"]
virus_cub_short.head()
```

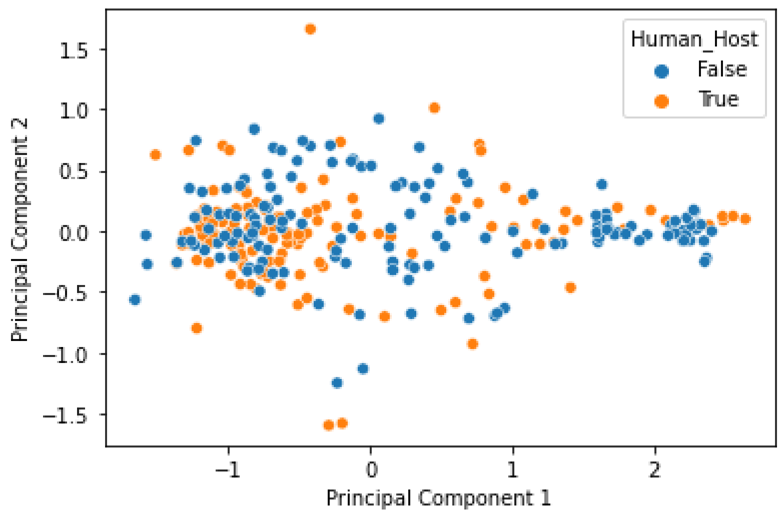
Out[13]:

	UUU	UUC	UUA	UUG	CUU	CUC	CUA	CUG	AU
39	0.138614	0.861386	0.009050	0.045249	0.167421	0.597285	0.027149	0.153846	0.113043
44	0.867647	0.132353	0.251701	0.217687	0.136054	0.027211	0.136054	0.231293	0.506173
45	0.950000	0.050000	0.541899	0.167598	0.078212	0.005587	0.117318	0.089385	0.500000
81	0.677966	0.322034	0.205036	0.273381	0.143885	0.079137	0.165468	0.133094	0.441860
105	0.779070	0.220930	0.236989	0.232342	0.298327	0.060409	0.095725	0.076208	0.533742

5 rows × 75 columns

In [14]:

```
# visualize virus_cub_short
virus_2d = reduce_dim(virus_cub_short, dim=2, drop_columns=viral_drop_columns)
virus_2d.head()
sns.scatterplot(data=virus_2d, x='Dim1', y='Dim2', hue='Human_Host')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2');
```



In [15]:
```
# dataset for our classifier, with just CUB and Human_Host columns

virus_cub_short_num = virus_cub_short.drop(columns=viral_drop_columns[:-1]) # ke
virus_cub_short_num.head()
```

Out[15]:

	UUU	UUC	UUA	UUG	CUU	CUC	CUA	CUG	AUL
39	0.138614	0.861386	0.009050	0.045249	0.167421	0.597285	0.027149	0.153846	0.113043
44	0.867647	0.132353	0.251701	0.217687	0.136054	0.027211	0.136054	0.231293	0.506173
45	0.950000	0.050000	0.541899	0.167598	0.078212	0.005587	0.117318	0.089385	0.500000
81	0.677966	0.322034	0.205036	0.273381	0.143885	0.079137	0.165468	0.133094	0.441860
105	0.779070	0.220930	0.236989	0.232342	0.298327	0.060409	0.095725	0.076208	0.533742

5 rows × 60 columns

and now, we build the classifier...

In [22]:
```
# separate data into test/train and validation sets
np.random.seed(88)
virus_valid, virus_tt = divide_data(virus_cub_short_num)
virus_tt.shape
```

Out[22]:
```
(278, 60)
```

In [17]:
```
# cross-validation for model selection

# get all cv errors for each method
cv_errors = all_cv_errors(virus_tt, ['neighbor', 'tree', 'forest', 'SVM'])

# store best method
best_method = cv_errors[0,0]
lowest_cv_err = cv_errors[0,1]

cv_errors
```

```
Out[17]: array([['tree', 0.24417989417989422],
                ['forest', 0.2736772486772487],
                ['neighbor', 0.2740740740740741],
                ['SVM', 0.4072751322751323]], dtype=object)
```

```
In [18]:
# print best model
print('best model')
print('method:   ', best_method)
print('error:    ', lowest_cv_err)
```

```
best model
method:   tree
error:    0.24417989417989422
```

```
In [19]:
# build best type of model
# mod_virus = KNeighborsClassifier(n_neighbors = 9)
# mod_virus = SVC(kernel="sigmoid")
mod_virus = DecisionTreeClassifier()
# mod_virus = RandomForestClassifier(n_estimators=20, max_features=min(3, virus_
mod_virus.fit(virus_tt.iloc[:,:-1], virus_tt.iloc[:,-1]) # from all but the last
```

```
Out[19]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                                max_depth=None, max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort='deprecated',
                                random_state=None, splitter='best')
```

```
In [20]:
# validate the model
# compute validation error
val_preds = mod_virus.predict(virus_valid.iloc[:,:-1])
val_error = classification_mse(val_preds, virus_valid.iloc[:,-1])
print('validation error:           ', val_error)
print('best cross-validation error:', lowest_cv_err)
```

```
validation error:            0.3
best cross-validation error: 0.24417989417989422
```

```
In [21]:
# confusion matrix
tp = np.sum((virus_valid.iloc[:,-1]) & (val_preds == 1))  # true positive
tn = np.sum(~(virus_valid.iloc[:,-1]) & (val_preds == 0))  # true negative
fp = np.sum(~(virus_valid.iloc[:,-1]) & (val_preds == 1)) # false positive  # ty
fn = np.sum((virus_valid.iloc[:,-1]) & (val_preds == 0)) # false negative   # ty
print('\tactual class')
print('   p\t\t   n')
print(f'tp: {tp/(tp+fn):.2f}\tfp: {fp/(tn+fp):.2f}')
print(f'fn: {fn/(tp+fn):.2f}\ttn: {tn/(tn+fp):.2f}')
```

```
        actual class
     p               n
tp: 0.53        fp: 0.08
fn: 0.47        tn: 0.92
```

In our validation set, we correctly predicted whether a virus is hosted by humans or not 70% of the time. Thus, our model performs moderately well, and certainly better than random chance. The moderate success of this classifier suggests that CUB may be a useful predictor of whether a virus may infect humans, alongside other factors. Potential applications could include incorporating CUB as a factor in models that predict emerging viral threats to the human species.