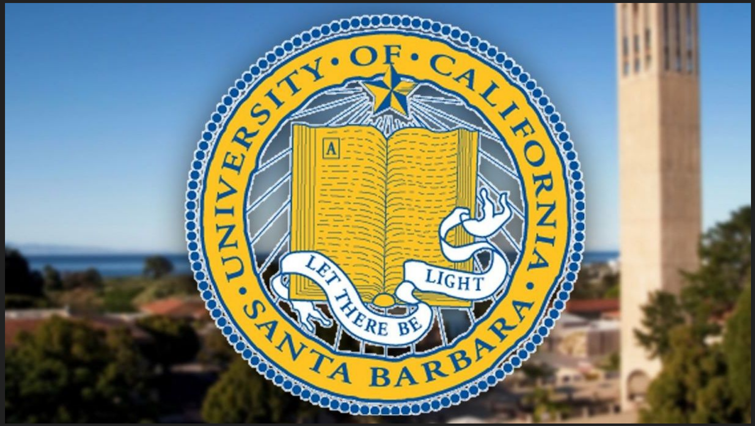
Predicting Cancerous p53 Mutants

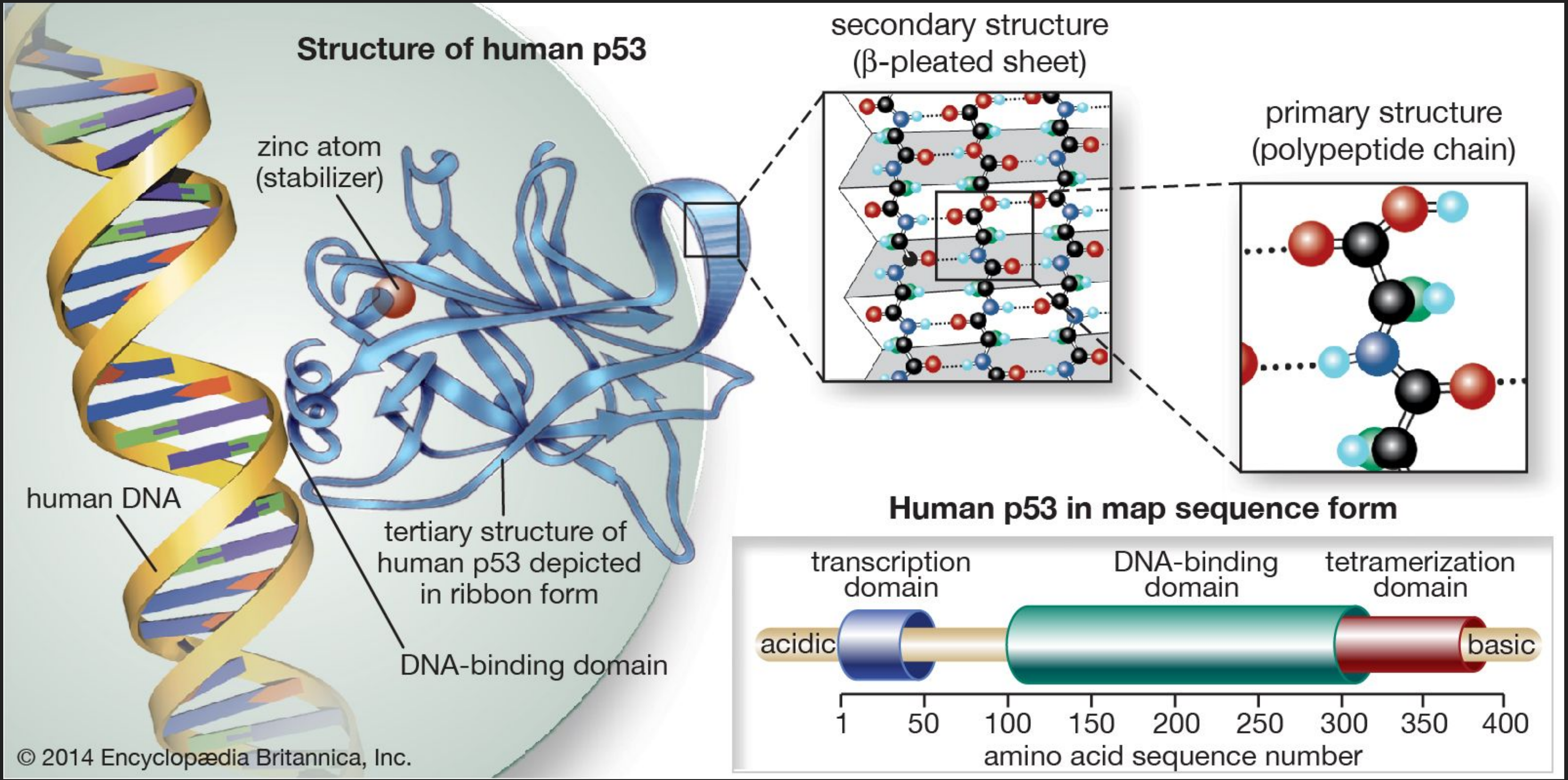
Shakuntala Mitra



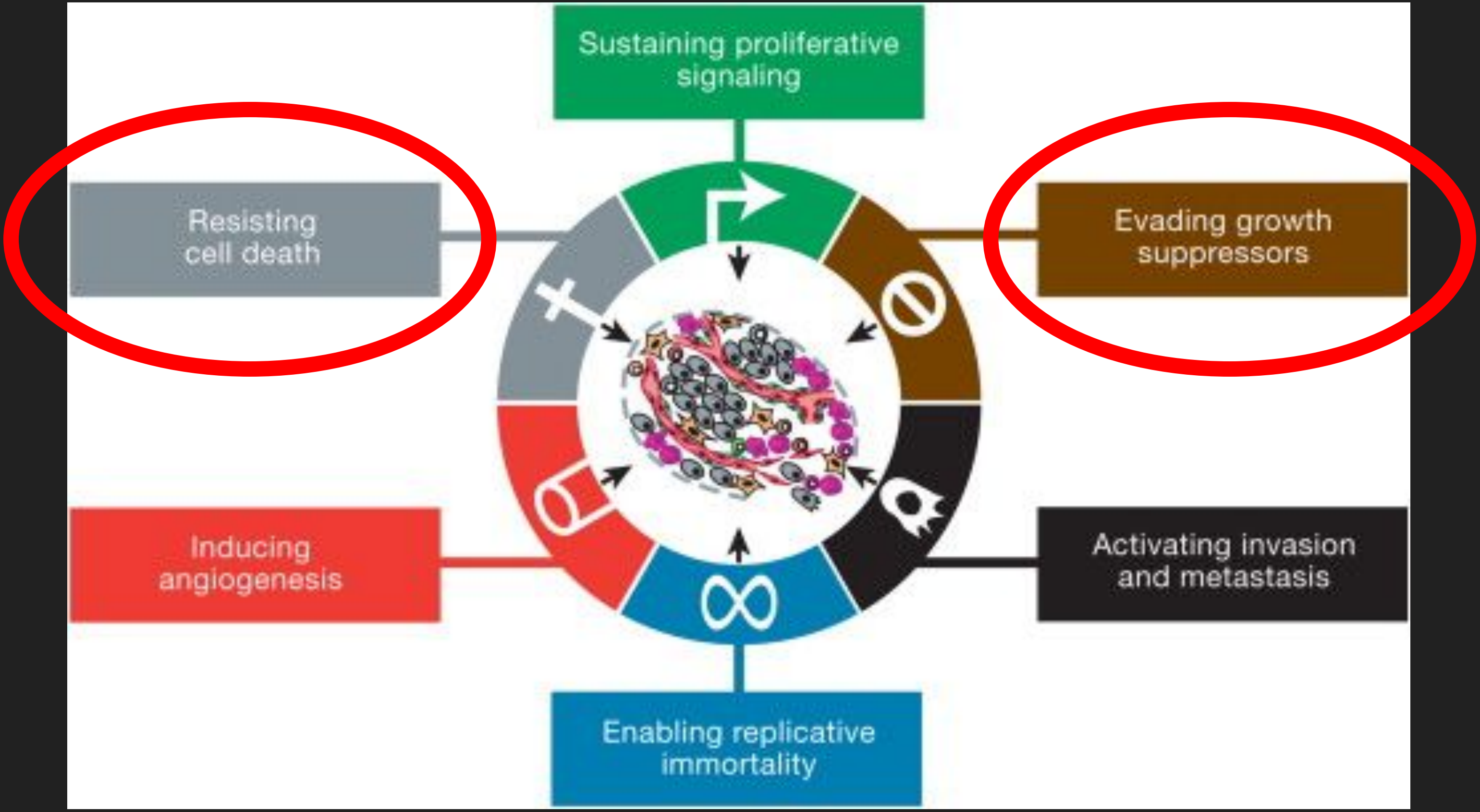
DATA SCIENCE UCSB

- B.S. Biochemistry & Molecular Biology
- Bioinformatics research
- Data Science UCSB Officer and Project Leader
- “Most Impactful Project” award at 2019 Annual Project Showcase

- CAR-T cell immunotherapies
- Data Science Certification, Advanced Machine Learning Specialization



p53 protein: The Guardian of the Cell



Hallmarks of Cancer (Hanahan 2011)

> 50%

Of all human cancers

What if there was a way to restore the normal function?

Classification (normal vs cancerous)

↓

Suppressor mutations (“corrective”)

↓

“Rescue” normal function

Data Landscape

- Source: UCI Machine Learning Repository
- Number of Instances: 16772
- Number of Attributes: 5409
- Attributes 1-4826: 2D electrostatic and surface based features.
- Attributes 4827-5408: 3D distance based features
- Attribute 5409 is the class attribute: active or inactive

```
print(k8_file.head())
```

```
     0       1       2       3       4       5      6      7      8     \
0  -0.161  -0.014   0.002  -0.036  -0.033  -0.093  0.025  0.005  0.000
1  -0.158  -0.002  -0.012  -0.025  -0.012  -0.106  0.013  0.005  0.000
2       ?       ?       ?       ?       ?       ?      ?      ?      ?
3  -0.169  -0.025  -0.010  -0.041  -0.045  -0.069  0.038  0.014  0.008
4  -0.183  -0.051  -0.023  -0.077  -0.092  -0.015  0.071  0.027  0.020

     9     ...    5400   5401   5402    5403    5404    5405   5406    5407  \
0  -0.015  ...   0.013  0.021   0.02   0.016  -0.011   0.003   0.01  -0.007
1  -0.002  ...  -0.008  0.007  0.015  -0.008  -0.011  -0.004  0.013   0.005
2       ?  ...       ?      ?      ?       ?       ?       ?      ?       ?
3  -0.014  ...    0.01  0.025  0.025   0.021  -0.012   0.006  0.016  -0.018
4  -0.019  ...   0.012   0.05  0.038   0.051  -0.015   0.017  0.027  -0.049

       5408 5409
0  inactive  NaN
1  inactive  NaN
2  inactive  NaN
3  inactive  NaN
4  inactive  NaN

[5 rows x 5410 columns]
```

- Missing Values
- Numeric Data
- Missing Labels
- High Dimensional
- Column of “NaN”

	index	0	1
0	1	a119e_l125p	inactive
1	2	a119e_r283k_a353v	inactive
2	3	a161t	inactive
3	4	c135y	inactive
4	5	c135y_e285m	inactive

- Separate File
- Mutant Nametags
- Also contains class attribute

Data Cleaning

- Concat numeric data and nametags
- Duplicate class attribute cols -> check equivalent -> drop one
- Change ‘?’ to NaN -> drop NaN rows -> no more missing values
- Check for duplicates -> None
- Check datatypes -> all “object”

1	2	3	4	5	6	7	8	9	...	5400	5401	5402	5403	5404	5405	5406	5407	5408	5409
-0.014	0.002	-0.036	-0.033	-0.093	0.025	0.005	0.000	-0.015	...	0.013	0.021	0.020	0.016	-0.011	0.003	0.010	-0.007	a119e_l125p	inactive
-0.002	-0.012	-0.025	-0.012	-0.106	0.013	0.005	0.000	-0.002	...	-0.008	0.007	0.015	-0.008	-0.011	-0.004	0.013	0.005	a119e_r283k_a353v	inactive
-0.025	-0.010	-0.041	-0.045	-0.069	0.038	0.014	0.008	-0.014	...	0.010	0.025	0.025	0.021	-0.012	0.006	0.016	-0.018	c135y	inactive
-0.051	-0.023	-0.077	-0.092	-0.015	0.071	0.027	0.020	-0.019	...	0.012	0.050	0.038	0.051	-0.015	0.017	0.027	-0.049	c135y_e285m	inactive
0.005	-0.011	-0.013	-0.002	-0.115	0.005	0.002	-0.003	0.002	...	0.012	0.009	0.003	-0.001	0.002	-0.006	0.009	0.013	c135y_e285v	inactive

```
singles.shape
```

(61, 5410)

```
doubles.shape
```

(16374, 5410)

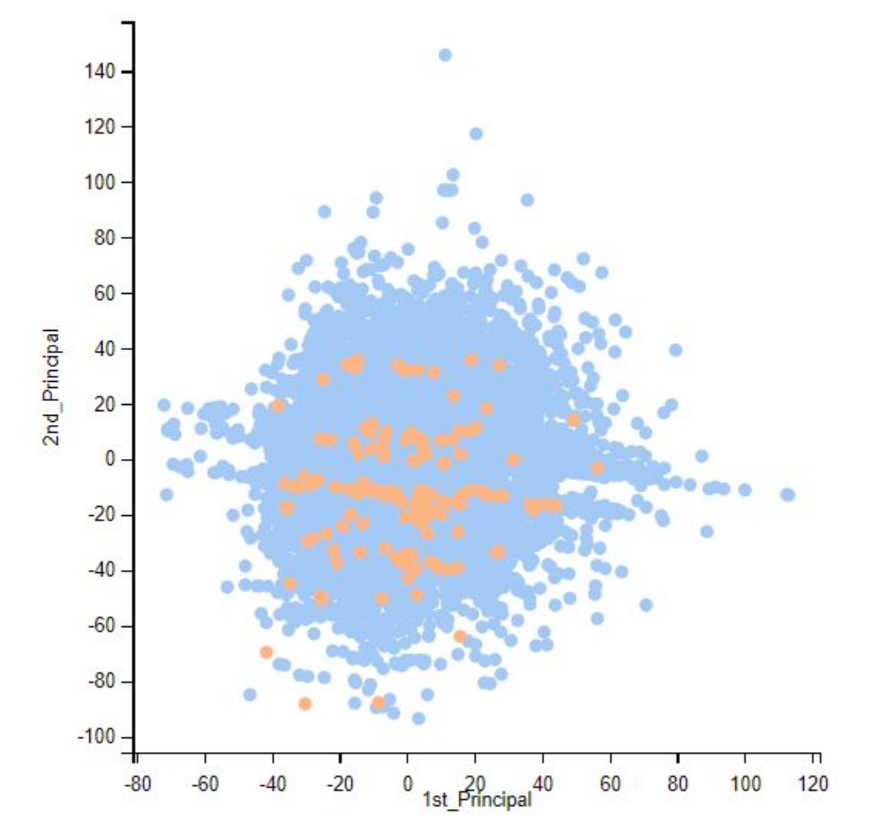
```
triples.shape
```

(114, 5410)

```
multis.shape # four or more mutations
```

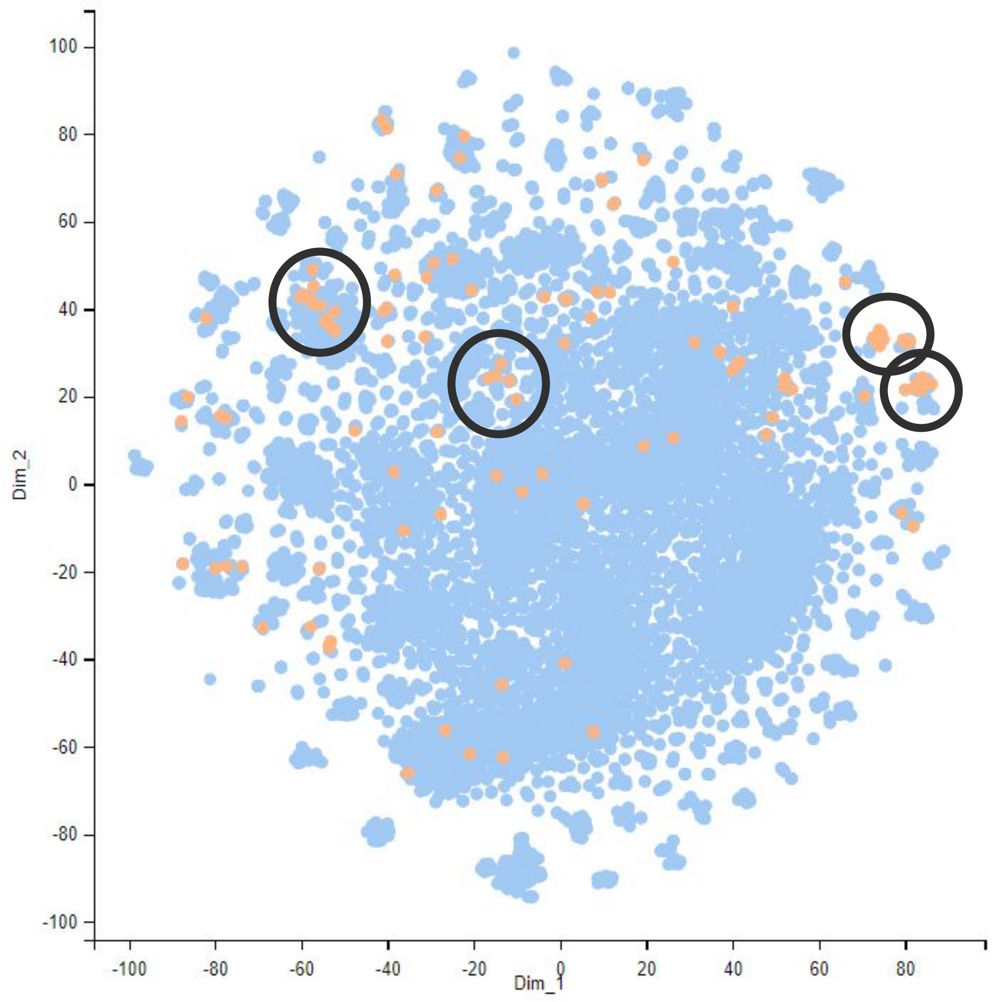
(42, 5410)

- Overwhelming majority have 2 mutations
- Four or more mutations is least common



- Large feature overlap
- Imbalanced classes
- Long clusters of “active” class

PCA Projection (2 Components): orange = active, blue = inactive (cancerous)



t-SNE

- Captures non-linear relationships better
- Still large feature overlap, better separation
- Scattered clusters of "active"

Time for preprocessing!

- All data must be numerical!
- Scale data
- Feature selection

```
0        False
1        False
2        False
3        False
4        False
         ...
16586     True
16587     True
16588    False
16589    False
16590     True
Name: 5409, Length: 16591, dtype: bool
```

Change Target Variable to Boolean

Feature Selection and Elimination

- Drop Features with No Variance or Low Variance
- Redundancy
  - Multicollinearity
  - Pointwise Mutual Information (PMI) Score

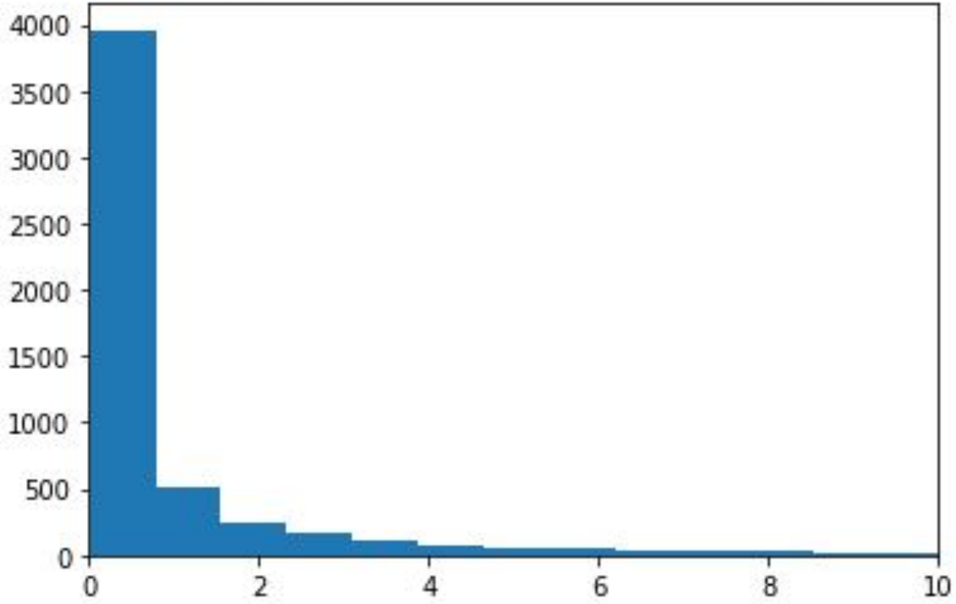
```
desc = mutants.describe()
desc.head()
```

	0	1	2	3	4	5	6
count	16591.000000	16591.000000	16591.000000	16591.000000	16591.000000	16591.000000	16591.000000
mean	-0.201763	-0.004898	-0.011593	-0.024726	-0.019615	-0.077004	0.222236
std	0.415089	0.362149	0.244942	0.256497	0.195207	0.539291	1.991421
min	-6.085000	-7.409000	-4.410000	-3.419000	-3.270000	-2.241000	-0.512000
25%	-0.169000	-0.024000	-0.014000	-0.040000	-0.043000	-0.113000	0.006000

5 rows × 5408 columns

Low Variance/No Variance

```
The minimum standard deviation for the original features is 0.006618814893812356
The maximum standard deviation for the original features is 77.3480716547277
```



Investigate Standard Deviations for Original Features

```
descT.describe() # focus on the 'std' column!
```

	count	mean	std	min	25%	50%	75%	max
count	5408.0	5408.000000	5408.000000	5408.000000	5408.000000	5408.000000	5408.000000	5408.000000
mean	16591.0	0.163703	1.188337	-10.564887	-0.275089	0.087606	0.564449	18.131332
std	0.0	3.482811	3.029671	16.820849	3.499720	3.806637	3.824114	31.861964
min	16591.0	-87.251644	0.006619	-255.926167	-91.454000	-86.965000	-82.645000	0.000000
25%	16591.0	-0.030156	0.148570	-11.160500	-0.061000	-0.025000	-0.004000	1.606750
50%	16591.0	0.013918	0.297502	-4.737000	-0.011000	0.011000	0.033000	4.345500
75%	16591.0	0.148052	0.892012	-2.339750	0.023000	0.135000	0.282000	19.545750
max	16591.0	57.955999	77.348072	-0.032000	47.499500	62.450000	71.864000	281.988000

Stats for Standard Deviations

```python
hv_mutants = mutants.drop(low_var_cols, axis=1)
hv_mutants.head()
```

	6	10	11	25	106	288	293	294	300	301	302	305	321	465	46
0	0.025	-0.030	-0.050	-0.031	-0.144053	-4.485	-4.222	-4.109	-0.033	-3.665	-2.676	-2.556	2.551	-0.079	0.0
1	0.013	-0.007	-0.010	-0.019	-0.172632	-4.489	-4.238	-4.111	-0.057	-3.725	-2.851	-2.645	-0.002	-0.063	0.0
2	0.038	-0.032	-0.043	-0.036	-0.158158	-4.485	-4.224	-4.107	-0.031	-3.643	-2.769	-2.580	-0.002	-0.094	0.0
3	0.071	-0.044	-0.097	-0.058	-0.038211	-4.484	-4.219	-4.106	-0.145	-3.998	-2.806	-2.557	-0.017	-0.142	0.0
4	0.005	0.006	-0.002	-0.011	-0.178263	-4.490	-4.239	-4.110	-0.060	-3.741	-2.841	-2.649	0.000	-0.049	0.0

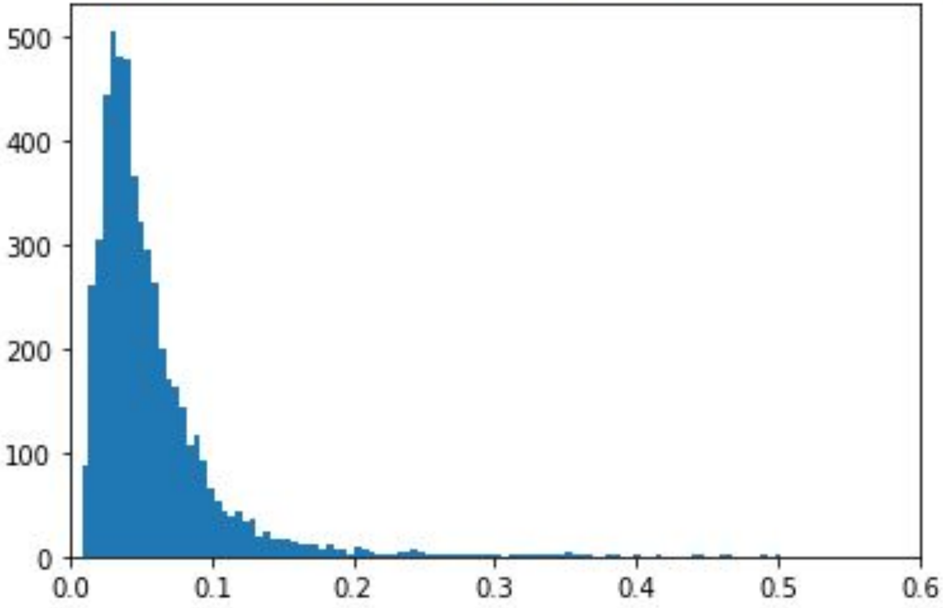
5 rows × 1259 columns

Remaining Features (Unscaled)

Scaling the Numerical Data is Important!

MaxAbsScaler (sklearn)

The minimum standard deviation for the Max Abs Scaled features is 0.007958482344045307
The maximum standard deviation for the Max Abs Scaled features is 0.502363693709944



Standard Deviations for Max Abs Scaled Features

```
mas_descT.describe()
```

	count	mean	std	min	25%	50%	75%	max
count	5408.0	5408.000000	5408.000000	5408.000000	5408.000000	5408.000000	5408.000000	5408.000000
mean	16591.0	0.003436	0.057177	-0.720603	-0.014358	0.001879	0.022439	0.763602
std	0.0	0.072722	0.046824	0.324982	0.087797	0.086291	0.071768	0.316891
min	16591.0	-0.626085	0.007958	-1.000000	-0.901205	-0.900000	-0.623126	0.000000
25%	16591.0	-0.007855	0.030296	-1.000000	-0.014298	-0.006195	-0.000910	0.522188
50%	16591.0	0.002711	0.044439	-0.906027	-0.002026	0.002046	0.006816	1.000000
75%	16591.0	0.017042	0.068514	-0.431275	0.003488	0.014670	0.031127	1.000000
max	16591.0	0.612498	0.502364	-0.008848	0.612200	0.718772	0.726974	1.000000

New Threshold for Low Variances (‘std’ < 0.1)

```python
# drop the features
MAS_mutants = mutants.drop(mas_low_var_cols, axis=1)
MAS_mutants.head()
```

	25	86	106	131	288	292	293	294	297	298	299	300	301	302	30
0	-0.031	-0.024	-0.144053	-0.021	-4.485	-0.694	-4.222	-4.109	0.003	-0.004	0.011	-0.033	-3.665	-2.676	0.
1	-0.019	0.006	-0.172632	-0.038	-4.489	-0.689	-4.238	-4.111	0.006	-0.002	-0.001	-0.057	-3.725	-2.851	0.
2	-0.036	-0.056	-0.158158	-0.033	-4.485	-0.696	-4.224	-4.107	-0.012	-0.017	0.001	-0.031	-3.643	-2.769	0.
3	-0.058	-0.116	-0.038211	-0.039	-4.484	-0.701	-4.219	-4.106	-0.038	-0.043	-0.011	-0.145	-3.998	-2.806	0.
4	-0.011	0.012	-0.178263	-0.037	-4.490	-0.688	-4.239	-4.110	0.011	0.004	0.002	-0.060	-3.741	-2.841	0.

5 rows × 555 columns

Remaining Features!

Linear Correlation-Based Feature Filtering

1. Calculate Linear Correlation of first two features with each other (redundancy)
   a. Higher than threshold -> collinear and redundant -> which to drop?
   b. Lower than threshold -> keep both features

2. Calculate average correlation of both features with rest of features (relevancy)

   a. Drop the feature with highest value (more repeated info)

3. Repeat process for all 2D features, then all 3D features

```
upper_2D = MAS_2D_corr.where(np.triu(np.ones(MAS_2D_corr.shape), k=1).astype(np.bool)) # make u
pper triangular matrix
upper_2D.head()
# ref: Chris Albon
```

	25	86	106	131	288	292	293	294	297	298	299
25	NaN	0.010741	0.013168	0.028168	0.024823	0.016069	0.024333	0.024354	0.009462	0.010356	0.010489
86	NaN	NaN	0.054569	0.045302	0.067544	0.009928	0.060062	0.056202	0.024092	0.020681	0.017643
106	NaN	NaN	NaN	0.020465	0.321220	0.194469	0.311122	0.307030	0.087466	0.089971	0.091480
131	NaN	NaN	NaN	NaN	0.057601	0.043589	0.059245	0.057614	0.021267	0.026795	0.032862
288	NaN	NaN	NaN	NaN	NaN	0.743613	0.991983	0.986423	0.464675	0.479323	0.479890

5 rows × 513 columns

Drop 200 2D features -> 313 of 2D features remaining

```python
lin_df = MAS_lin_comb.rename({5409: 'activity', 5408: 'nametags'}, axis='columns')
lin_df.head()
```

	25	86	131	294	297	307	321	340	345	346	370	500	506	536	553
0	-0.031	-0.024	-0.021	-4.109	0.003	-0.103000	2.551	3.024	-0.442	-0.001	1.038	0.048	-0.019	0.033	0.04
1	-0.019	0.006	-0.038	-4.111	0.006	-0.054667	-0.002	2.952	-0.366	0.009	0.992	0.001	-0.002	0.031	0.01
2	-0.036	-0.056	-0.033	-4.107	-0.012	-0.099333	-0.002	3.031	-0.089	-0.019	1.051	0.041	-0.024	0.042	-1.1
3	-0.058	-0.116	-0.039	-4.106	-0.038	-0.134333	-0.017	-0.033	-0.056	-0.046	1.180	4.554	4.426	0.063	-1.0
4	-0.011	0.012	-0.037	-4.110	0.011	-0.057333	0.000	2.974	-0.362	0.017	0.949	0.010	0.005	0.025	0.00

5 rows × 344 columns

Drop 13 3D features -> combine -> 343 Remaining Features

Pairwise Mutual Information-Based Feature Filtering

1. Calculate PMI score of first two features with each other (redundancy)
   a. Higher than threshold -> collinear and redundant -> which to drop?
   b. Lower than threshold -> keep both features

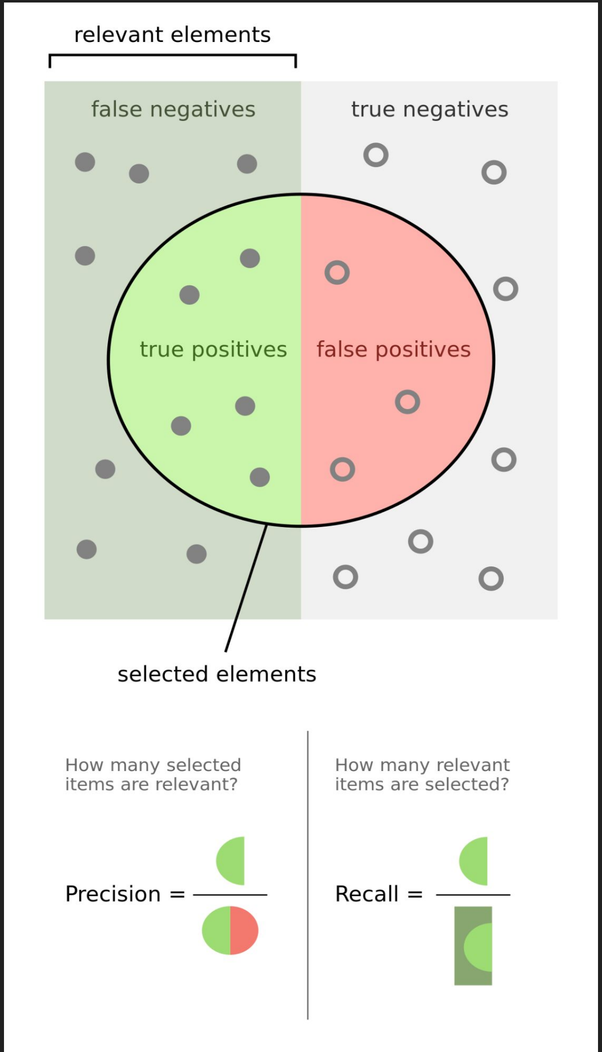
2. Calculate average entropy of both features with rest of features (relevancy)

   a. Drop feature with lowest entropy (very predictable, low information gain)

3. Repeat process for all 2D features, then all 3D features

What are our metrics for success?

Recall is more important than precision!



Evaluating Baseline Models

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
NearestCentroid	0.79	0.83	0.83	0.88	0.26
GaussianNB	0.71	0.77	0.77	0.83	0.31
LinearDiscriminantAnalysis	0.98	0.72	0.72	0.99	1.41
BernoulliNB	0.69	0.67	0.67	0.81	0.36
XGBClassifier	0.99	0.66	0.66	0.99	27.80
LinearSVC	0.99	0.66	0.66	0.99	3.23
DecisionTreeClassifier	0.98	0.63	0.63	0.99	14.85
KNeighborsClassifier	0.99	0.61	0.61	0.99	29.40
AdaBoostClassifier	0.99	0.61	0.61	0.99	23.33
LogisticRegression	0.99	0.61	0.61	0.99	1.22
ExtraTreeClassifier	0.99	0.61	0.61	0.99	0.25
ExtraTreesClassifier	0.99	0.59	0.59	0.99	3.07
BaggingClassifier	0.99	0.59	0.59	0.99	87.84
PassiveAggressiveClassifier	0.99	0.59	0.59	0.99	0.41
LGBMClassifier	0.99	0.57	0.57	0.99	10.41
LabelSpreading	0.99	0.57	0.57	0.99	12.48
LabelPropagation	0.99	0.57	0.57	0.99	9.04
SGDClassifier	0.99	0.54	0.54	0.99	3.77
RandomForestClassifier	0.99	0.52	0.52	0.99	28.60
Perceptron	0.99	0.52	0.52	0.99	0.45
QuadraticDiscriminantAnalysis	0.99	0.50	0.50	0.99	1.05
RidgeClassifier	0.99	0.50	0.50	0.99	0.43
RidgeClassifierCV	0.99	0.50	0.50	0.99	1.08
SVC	0.99	0.50	0.50	0.99	9.23
CalibratedClassifierCV	0.99	0.50	0.50	0.99	9.28
DummyClassifier	0.98	0.49	0.49	0.98	0.21

Balanced Accuracy and Naive Classifiers

$$\text{Balanced accuracy} = \frac{\text{Sensitivity} + \text{Specificity}}{2}.$$

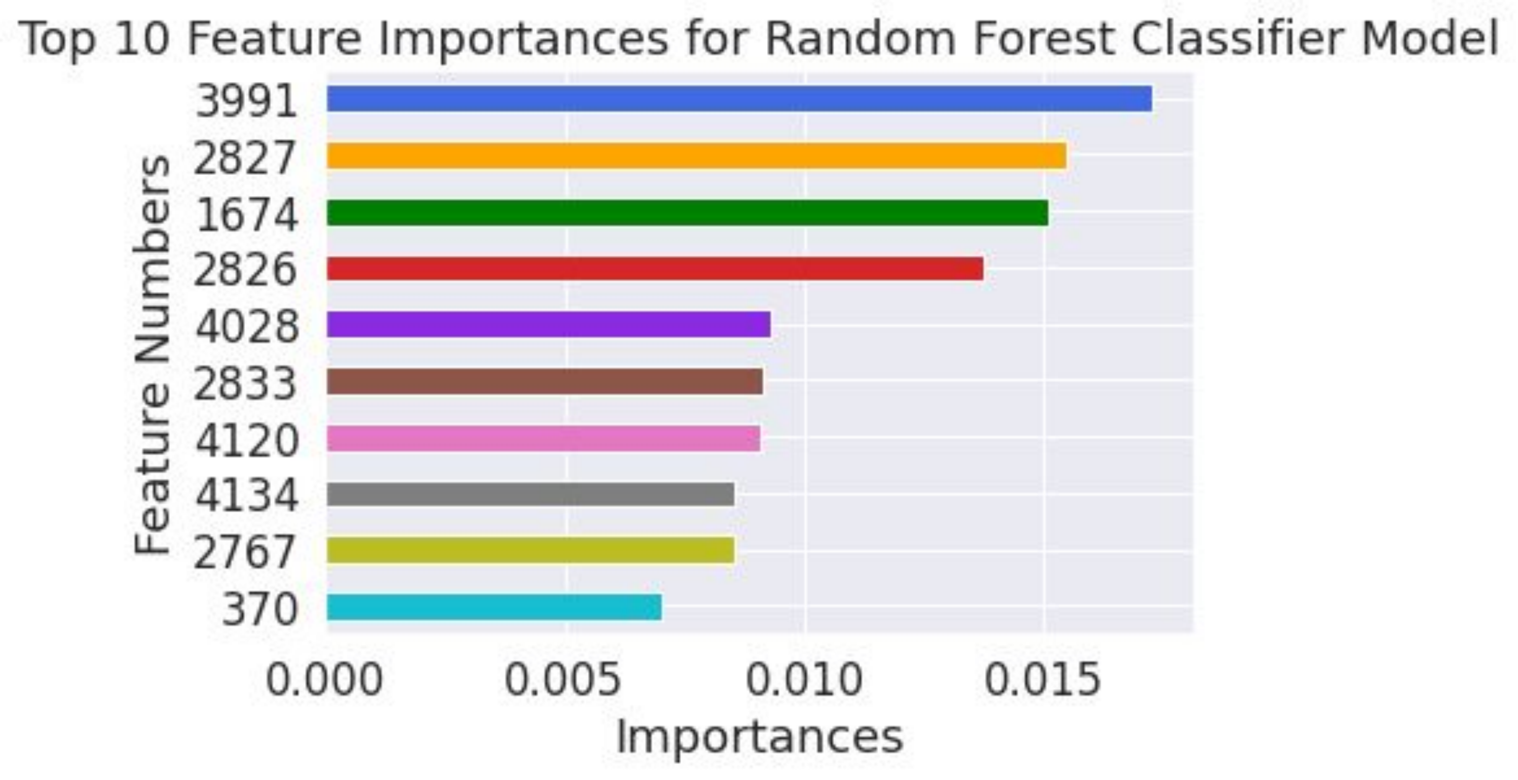
- Naive classifier = always predicts majority class
  - Balanced accuracy = 0.50 (50%)
- Want balanced accuracy > 0.50

	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
Model					
NearestCentroid	0.79	0.83	0.83	0.88	0.26
GaussianNB	0.71	0.77	0.77	0.83	0.31
LinearDiscriminantAnalysis	0.98	0.72	0.72	0.99	1.41
BernoulliNB	0.69	0.67	0.67	0.81	0.36
XGBClassifier	0.99	0.66	0.66	0.99	27.80
LinearSVC	0.99	0.66	0.66	0.99	3.23
DecisionTreeClassifier	0.98	0.63	0.63	0.99	14.85
KNeighborsClassifier	0.99	0.61	0.61	0.99	29.40
AdaBoostClassifier	0.99	0.61	0.61	0.99	23.33
LogisticRegression	0.99	0.61	0.61	0.99	1.22
ExtraTreeClassifier	0.99	0.61	0.61	0.99	0.25
ExtraTreesClassifier	0.99	0.59	0.59	0.99	3.07
BaggingClassifier	0.99	0.59	0.59	0.99	87.84
PassiveAggressiveClassifier	0.99	0.59	0.59	0.99	0.41
LGBMClassifier	0.99	0.57	0.57	0.99	10.41
LabelSpreading	0.99	0.57	0.57	0.99	12.48
LabelPropagation	0.99	0.57	0.57	0.99	9.04
SGDClassifier	0.99	0.54	0.54	0.99	3.77
RandomForestClassifier	0.99	0.52	0.52	0.99	28.60
Perceptron	0.99	0.52	0.52	0.99	0.45
QuadraticDiscriminantAnalysis	0.99	0.50	0.50	0.99	1.05
RidgeClassifier	0.99	0.50	0.50	0.99	0.43
RidgeClassifierCV	0.99	0.50	0.50	0.99	1.08
SVC	0.99	0.50	0.50	0.99	9.23
CalibratedClassifierCV	0.99	0.50	0.50	0.99	9.28
DummyClassifier	0.98	0.49	0.49	0.98	0.21

Chosen Models

- Nearest Centroid
- Gaussian Naive Bayes
- Random Forest Classifier
- Logistic Regression Classifier

Random Forest v1



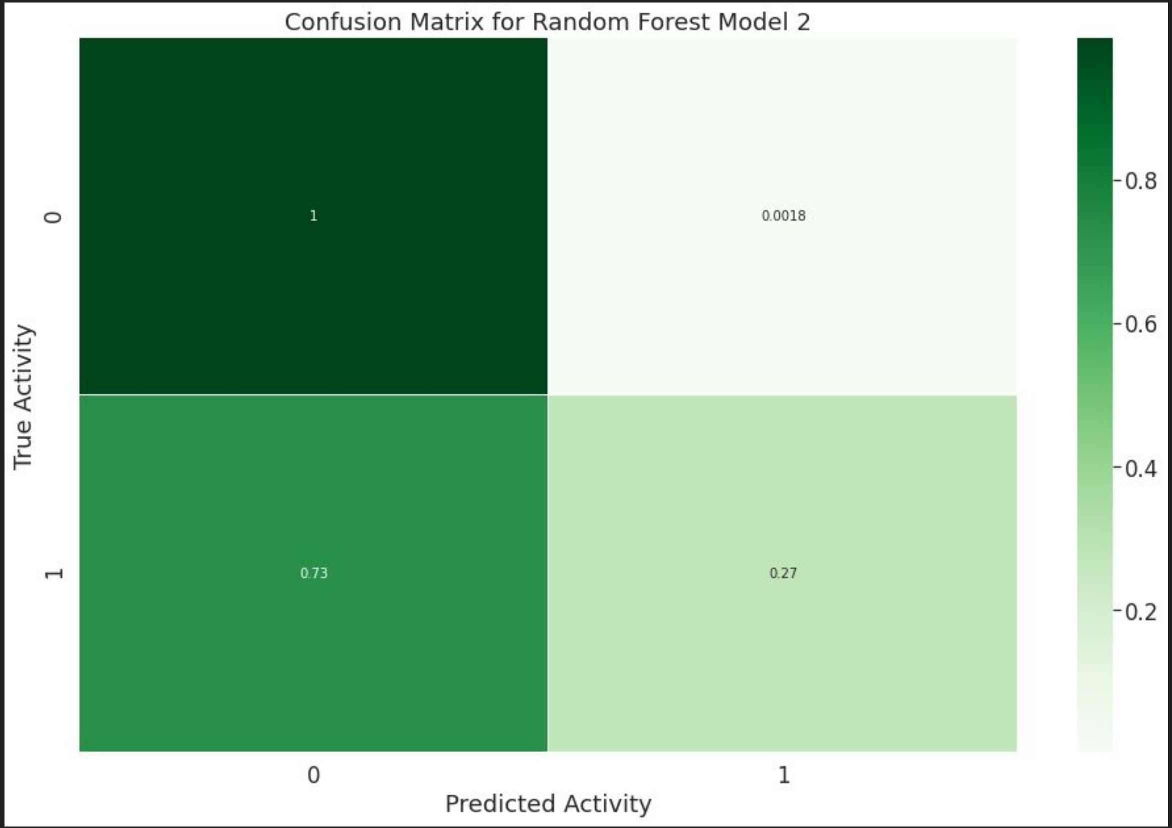
Top 10 Feature Importances for Random Forest Classifier Model
Feature Numbers
3991
2827
1674
2826
4028
2833
4120
4134
2767
370
0.000
0.005
0.010
0.015
Importances


Class Balance of Original Training Data
Classes
True
False
0
2000
4000
6000
8000
10000
12000
Number of Observations
Class Balance of Resampled Training Data
Classes
False
True
0
2000
4000
6000
8000
10000
12000
Number of Observations

SMOTE-Tomek

	precision	recall	f1-score	support
False	1.00	1.00	1.00	3297
True	0.50	0.27	0.35	22
accuracy			0.99	3319
macro avg	0.75	0.64	0.67	3319
weighted avg	0.99	0.99	0.99	3319

Random Forest v2



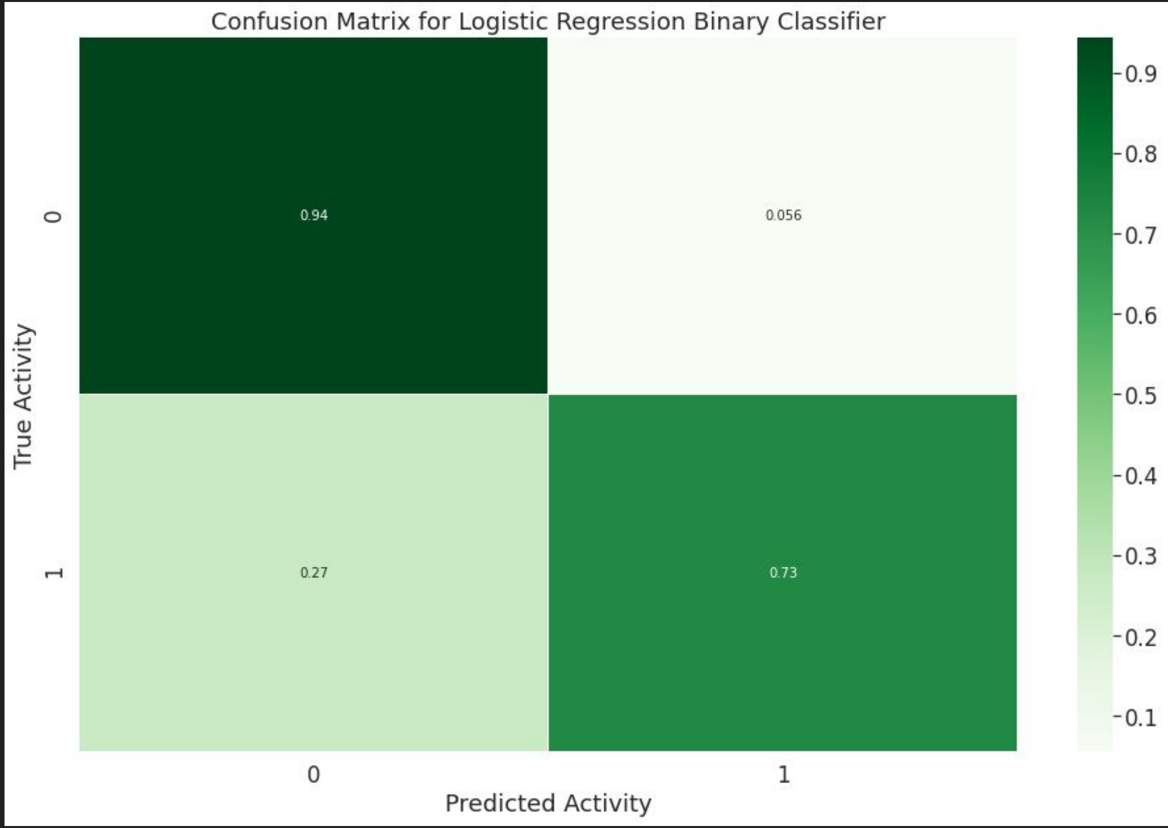
Random Forest v2

	precision	recall	f1-score	support
False	1.00	0.99	0.99	3297
True	0.24	0.27	0.26	22
accuracy			0.99	3319
macro avg	0.62	0.63	0.63	3319
weighted avg	0.99	0.99	0.99	3319

Random Forest v3 (Increasing Weights on Minority Class)

	precision	recall	f1-score	support
False	0.99	1.00	0.99	3297
True	0.17	0.14	0.15	22
accuracy			0.99	3319
macro avg	0.58	0.57	0.57	3319
weighted avg	0.99	0.99	0.99	3319

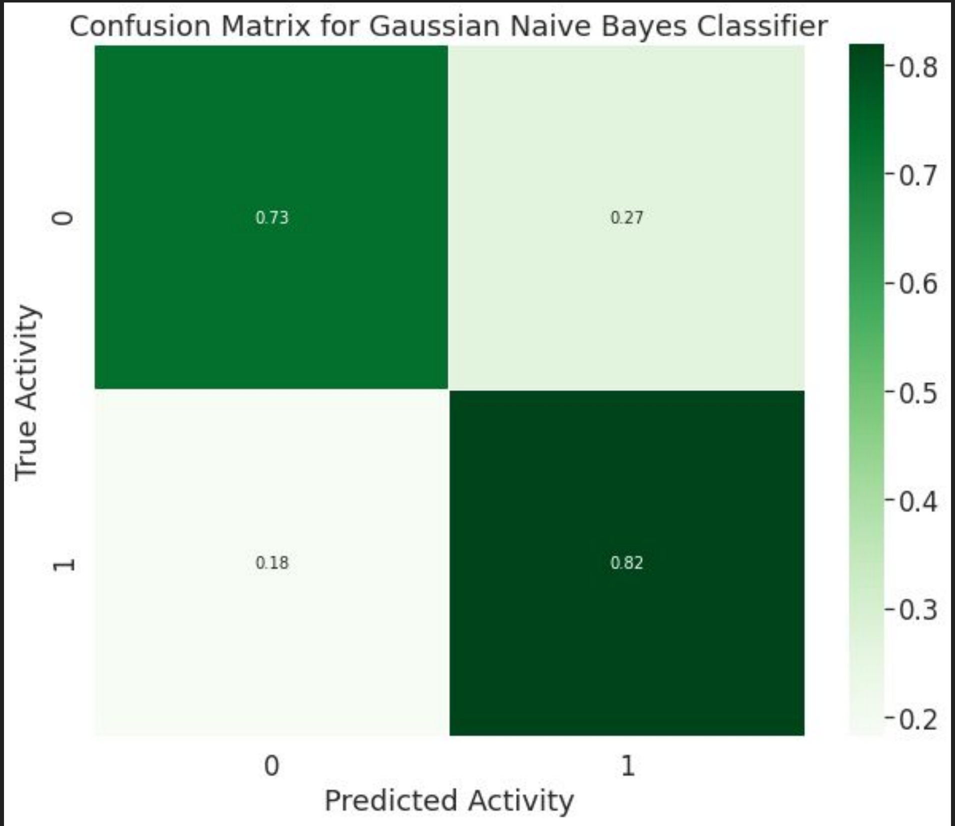
Random Forest with Randomized Search



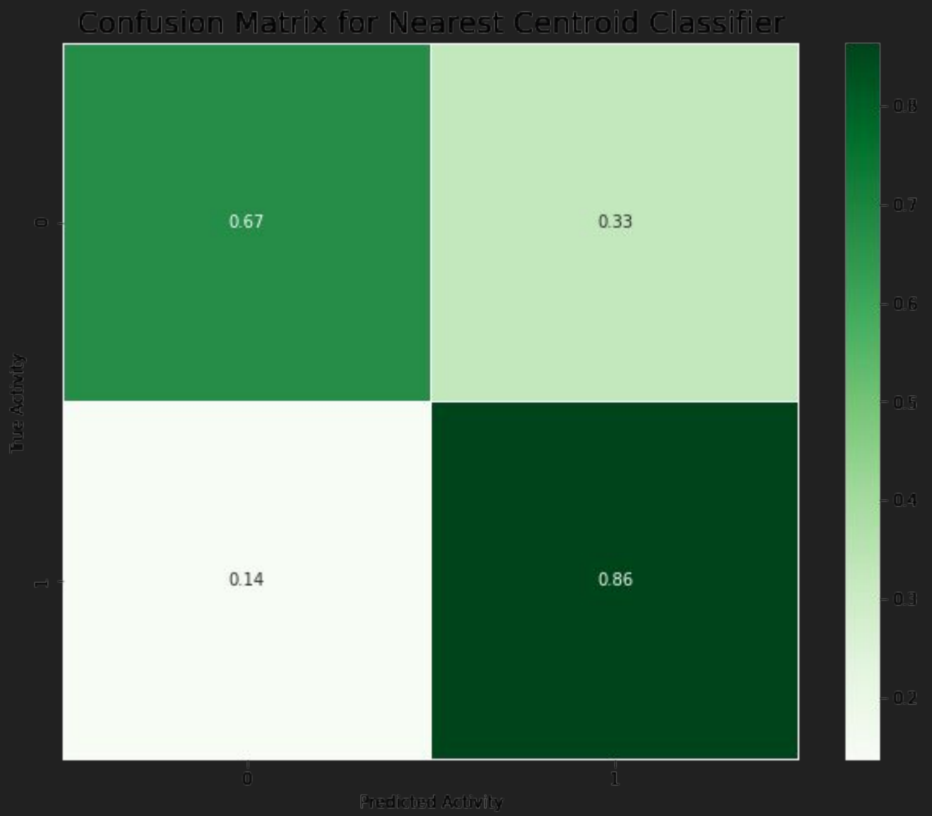
Logistic Regression

	precision	recall	f1-score	support
False	1.00	0.94	0.97	3297
True	0.08	0.73	0.14	22
accuracy			0.94	3319
macro avg	0.54	0.84	0.56	3319
weighted avg	0.99	0.94	0.96	3319

Logistic Regression



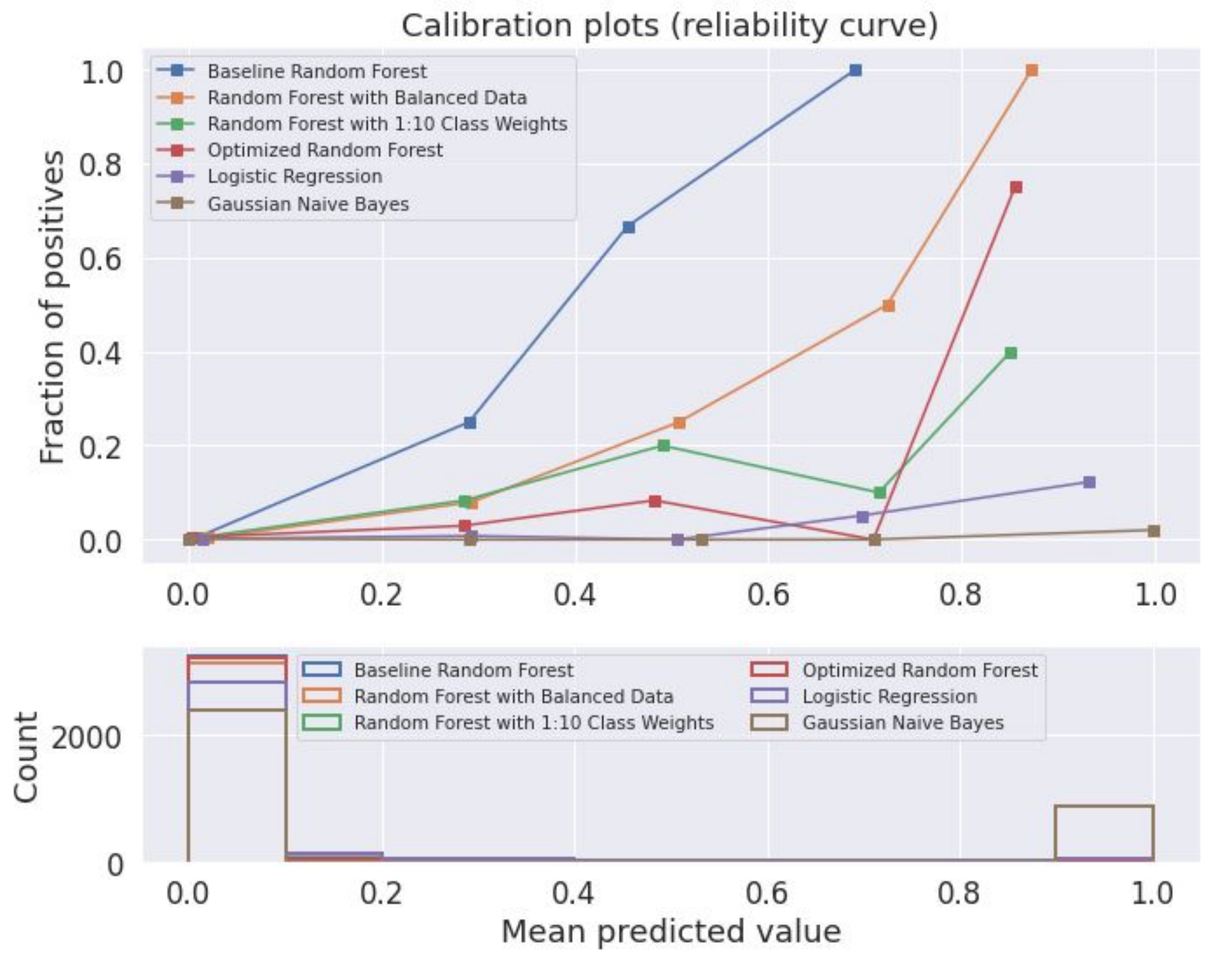
Gaussian Naive Bayes



Nearest Centroid

	precision	recall	f1-score	support
False	1.00	0.67	0.81	3297
True	0.02	0.86	0.03	22
accuracy			0.68	3319
macro avg	0.51	0.77	0.42	3319
weighted avg	0.99	0.68	0.80	3319

Nearest Centroid


Calibration plots (reliability curve)
Baseline Random Forest
Random Forest with Balanced Data
Random Forest with 1:10 Class Weights
Optimized Random Forest
Logistic Regression
Gaussian Naive Bayes
Fraction of positives
1.0
0.8
0.6
0.4
0.2
0.0
Count
2000
0
Mean predicted value

Discussion: Best model (so far)

- Nearest Centroid had best recall for minority class
  - 86%
- High recall, low precision
- Reliability curves - not ideal

Questions to Think About

- Out of mutants predicted to be non-cancerous
  - Affected domain(s)? (DBD, NLS, etc)
  - Core domain affected?
- Potentially visualize “active” mutants with PyMol

Future Work and Improvements, pt.1

- Filtering Features Using Pairwise Mutual Information
- Play with the number of features/attributes used to train the models to find optimum number of features
- Instead of just a train-test-split, make a train-test-validation split of the data
- Visualize the data with UMAP and Compare UMAP with t-SNE
- Hyperparameter Tuning with Bayesian Optimization
- Resampling Data with SMOTE-ENN and comparing noise with SMOTE-Tomek

Future Work and Improvements, pt.2

- Calibrating the Classifier Models
- Neural Networks
- Calculate the MCC scores
- This was built using the 2010 dataset, but can combine with 2012 dataset
- Investigate specific clusters of “active” p53 proteins more closely
- Combine with protein visualization software for easier interpretation of the results
    - Maybe cross-check/sanity check with which domains of p53 are the most important to preserving wild-type function
- Use Cloud Computing services, containerize and deploy model