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

In [2]:
```
biomarkers_df = pd.read_csv("/Users/akkaedoodle/Downloads/debernardi_f
```

In [3]:
```
biomarkers_df
```

Out[3]:

	sample_id	patient_cohort	sample_origin	age	sex	diagnosis	stage	benign_sample_diag
0	S1	Cohort1	BPTB	33	F	1	-1	
1	S100	Cohort2	BPTB	51	M	1	-1	
2	S101	Cohort2	BPTB	61	M	1	-1	
3	S102	Cohort2	BPTB	62	M	1	-1	
4	S105	Cohort2	BPTB	58	F	1	-1	
...	...	...	...	...	...	...	...	...
237	S484	Cohort1	BPTB	44	F	3	4	
238	S495	Cohort1	BPTB	58	M	3	4	
239	S519	Cohort1	BPTB	78	F	3	4	
240	S529	Cohort1	BPTB	61	F	3	4	
241	S590	Cohort1	BPTB	74	M	3	4	

242 rows × 18 columns

In [4]:
```
biomarkers_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 242 entries, 0 to 241
Data columns (total 18 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   sample_id                 242 non-null    object
```

In [5]:
```
sex = {'F':0, 'M':1}
biomarkers_df.sex = [sex[item] for item in biomarkers_df.sex]
```

In [6]:
```
from sklearn.model_selection import train_test_split
```

Drop Stuff

In [7]:
```
biomarkers_df.drop('sample_id', axis=1, inplace=True)
biomarkers_df.drop('patient_cohort', axis=1, inplace=True)
biomarkers_df.drop('sample_origin', axis=1, inplace=True)
biomarkers_df.drop('benign_sample_diagnosis', axis=1, inplace=True)
#biomarkers_df.drop('plasma_CA19_9', axis=1, inplace=True)
biomarkers_df.drop('REG1A', axis=1, inplace=True)
```

In [8]:
```
biomarkers_df.drop('sex', axis=1, inplace=True)
biomarkers_df.drop('age', axis=1, inplace=True)
```

In [9]:
```
biomarkers_df.drop('creatinine_log', axis=1, inplace=True)
biomarkers_df.drop('LYVE_log', axis=1, inplace=True)
biomarkers_df.drop('REG1B_log', axis=1, inplace=True)
biomarkers_df.drop('TFF1_log', axis=1, inplace=True)
biomarkers_df.drop('TFF1', axis=1, inplace=True)
```

In [10]:
```
biomarkers_df.drop('stage', axis=1, inplace=True)
```

In [11]:
```
biomarkers_df
```

Out[11]:

	diagnosis	plasma_CA19_9	creatinine	LYVE1	REG1B
0	1	11.7	1.83222	0.893219	52.948840
1	1	7.0	0.78039	0.145589	102.366000
2	1	8.0	0.70122	0.002805	60.579000
3	1	9.0	0.21489	0.000860	65.540000
4	1	11.0	0.89349	0.003574	3.730000
...	...	...	...	...	...
237	3	271.7	2.42034	9.005338	144.985040
238	3	710.8	0.19227	3.055294	32.890960
239	3	941.0	0.46371	1.044345	14.364360
240	3	13740.0	0.32799	5.232527	123.104730
241	3	1488.0	1.50423	8.200958	411.938275

242 rows × 5 columns

Decision Tree Classifier

In [12]:
```
X = biomarkers_df.drop('diagnosis', axis=1)
y = biomarkers_df['diagnosis']
```

In [13]:
```
X.describe()
```

Out[13]:

	plasma_CA19_9	creatinine	LYVE1	REG1B
count	242.000000	242.000000	242.000000	242.000000
mean	918.298224	0.845025	3.640866	141.743829
std	2881.200056	0.691156	3.892657	218.922695
min	0.000000	0.067860	0.000129	0.730367
25%	7.000000	0.327990	0.033969	13.651235
50%	63.000000	0.633360	2.475677	56.139567
75%	555.500000	1.114035	5.817225	168.084348
max	31000.000000	4.116840	23.890323	1215.168000

In [14]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
```

In [15]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [16]:
```
dtree = DecisionTreeClassifier()
```

In [17]:
```
dtree.fit(X_train,y_train)
```

Out[17]:
```
DecisionTreeClassifier()
```

In [18]:
```
predictions = dtree.predict(X_test)
```

In [19]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

In [20]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           1       0.97      0.85      0.91        40
           3       0.93      0.99      0.96        81

    accuracy                           0.94       121
   macro avg       0.95      0.92      0.93       121
weighted avg       0.94      0.94      0.94       121
```

In [21]:
```
print(confusion_matrix(y_test,predictions))
```

```
[[34  6]
 [ 1 80]]
```

Random Forest Classifier

In [22]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, y_train)
```

Out[22]:
```
RandomForestClassifier()
```

In [23]:
```
rfc_pred = rfc.predict(X_test)
```

In [24]:
```
print(confusion_matrix(y_test,rfc_pred))
```

```
[[37  3]
 [ 3 78]]
```

In [25]:
```
print(classification_report(y_test,rfc_pred))
```

```
              precision    recall  f1-score   support

           1       0.93      0.93      0.93        40
           3       0.96      0.96      0.96        81

    accuracy                           0.95       121
   macro avg       0.94      0.94      0.94       121
weighted avg       0.95      0.95      0.95       121
```

In [26]:
```
y.value_counts()
```

Out[26]:
```
3    150
1     92
Name: diagnosis, dtype: int64
```

In [27]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state
```

In [28]:
```
X_train.describe()
```

Out[28]:

	plasma_CA19_9	creatinine	LYVE1	REG1B
count	181.000000	181.000000	181.000000	181.000000
mean	930.577441	0.825942	3.607254	133.452653
std	2965.269471	0.695318	3.955940	216.143107
min	0.000000	0.067860	0.000129	0.954900
25%	6.940000	0.327990	0.016368	13.027030

	plasma_CA19_9	creatinine	LYVE1	REG1B
50%	66.000000	0.576810	2.236909	47.066000
75%	556.000000	1.063140	5.875759	167.323310

In [29]:
```
forest = RandomForestClassifier()
```

In [30]:
```
forest.fit(X_train, y_train)
```

Out[30]:
```
RandomForestClassifier()
```

In [31]:
```
y_pred_test = forest.predict(X_test)
```

In [32]:
```
y_pred_test
```

Out[32]:
```
array([3, 1, 3, 3, 3, 1, 1, 3, 3, 1, 3, 1, 3, 3, 3, 3, 3, 3, 3, 1, 3,
3,
       3, 1, 3, 3, 3, 3, 1, 3, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3,
3,
       1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 1, 1, 3, 3, 3, 3, 1])
```

In [33]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, classifi
```

In [34]:
```
accuracy_score(y_test, y_pred_test)
```

Out[34]:
```
0.8688524590163934
```

random forest Confusion Matrix

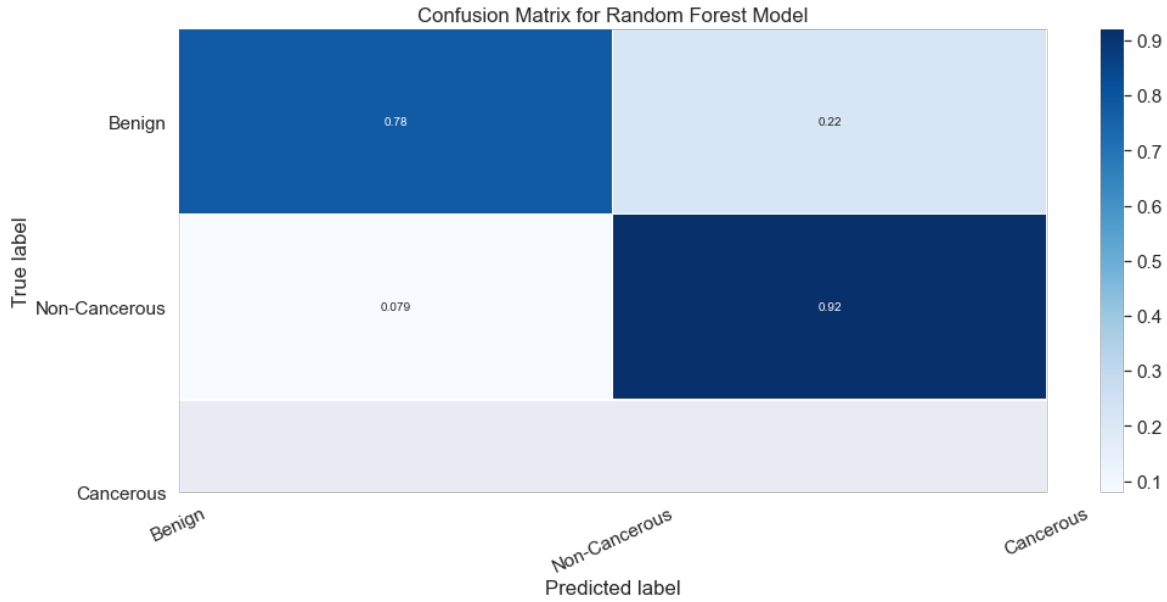
In [35]:
```
confusion_matrix(y_test, y_pred_test)
```

Out[35]:
```
array([[18,  5],
       [ 3, 35]])
```

In [36]:
```
matrix = confusion_matrix(y_test, y_pred_test)
matrix = matrix.astype('float') / matrix.sum(axis=1)[:, np.newaxis]

plt.figure(figsize=(16,7))
sns.set(font_scale=1.4)
sns.heatmap(matrix, annot=True, annot_kws={'size':10}, cmap=plt.cm.Blu

class_names = ['Benign', 'Non-Cancerous', 'Cancerous']
tick_marks = np.arange(len(class_names))
tick_marks2 = tick_marks + 0.5
plt.xticks(tick_marks, class_names, rotation=25)
plt.yticks(tick_marks2, class_names, rotation=0)
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix for Random Forest Model')
plt.show()
```



Support Vector Classifier

In [37]:
```
from sklearn.svm import SVC
```

In [38]:
```
model = SVC()
```

In [39]:
```
model.fit(X_train, y_train)
```

Out[39]:
```
SVC()
```

In [40]:
```
predictions = model.predict(X_test)
```

In [41]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [42]:
```
print(confusion_matrix(y_test,predictions))
```

```
[[23  0]
 [15 23]]
```

In [43]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           1       0.61      1.00      0.75        23
           3       1.00      0.61      0.75        38

    accuracy                           0.75        61
   macro avg       0.80      0.80      0.75        61
weighted avg       0.85      0.75      0.75        61
```

SVC improved drastically after removing all the actual protein/creatinine values (and only keeping the logs)

In [44]:
```
param_grid = {'C': [0.1,1, 10, 100, 1000], 'gamma': [1,0.1,0.01,0.001,
```

In [45]:
```
from sklearn.model_selection import GridSearchCV
```

In [46]:
```
grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=3)
```

In [47]:
```
grid.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
[CV 1/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.622 total t
ime=   0.0s
[CV 2/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.611 total t
ime=   0.0s
[CV 3/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.611 total t
ime=   0.0s
[CV 4/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.611 total t
ime=   0.0s
[CV 5/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.639 total t
ime=   0.0s
[CV 1/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.622 total t
ime=   0.0s
[CV 2/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.639 total t
ime=   0.0s
[CV 3/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.611 total t
ime=   0.0s
[CV 4/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.611 total t
ime=   0.0s
```

In [48]:
```
grid.best_params_
```

Out[48]:
```
{'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}
```

In [49]:
```
grid.best_estimator_
```

Out[49]:
```
SVC(C=10, gamma=0.0001)
```

In [50]:
```
grid_predictions = grid.predict(X_test)
```

In [51]:
```
print(confusion_matrix(y_test,grid_predictions))
```

```
[[18  5]
 [ 1 37]]
```

In [52]:
```
print(classification_report(y_test,grid_predictions))
```

```
              precision    recall  f1-score   support

           1       0.95      0.78      0.86        23
           3       0.88      0.97      0.93        38

    accuracy                           0.90        61
   macro avg       0.91      0.88      0.89        61
weighted avg       0.91      0.90      0.90        61
```

Logistic Regression

In [53]:
```
from sklearn.linear_model import LogisticRegression
```

In [54]:
```
classifier = LogisticRegression(random_state = 1)
```

In [55]:
```
classifier.fit(X_train, y_train)
```

Out[55]:
```
LogisticRegression(random_state=1)
```

In [56]:
```
y_train.describe()
```

Out[56]:
```
count    181.000000
mean       2.237569
std        0.974065
min        1.000000
25%        1.000000
50%        3.000000
75%        3.000000
max        3.000000
Name: diagnosis, dtype: float64
```

In [57]:
```
X_train.describe()
```

Out[57]:

	plasma_CA19_9	creatinine	LYVE1	REG1B
count	181.000000	181.000000	181.000000	181.000000
mean	930.577441	0.825942	3.607254	133.452653
std	2965.269471	0.695318	3.955940	216.143107

	plasma_CA19_9	creatinine	LYVE1	REG1B
min	0.000000	0.067860	0.000129	0.954900
25%	6.940000	0.327990	0.016368	13.027030
50%	66.000000	0.576810	2.236909	47.066000
75%	556.000000	1.063140	5.875759	167.323310

In [58]:
```
Y_pred = classifier.predict(X_test)
```

In [59]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, Y_pred)
```

In [60]:
```
cm
```

Out[60]:
```
array([[18,  5],
       [ 2, 36]])
```

In [61]:
```
print(classification_report(y_test,Y_pred))
```

```
              precision    recall  f1-score   support

           1       0.90      0.78      0.84        23
           3       0.88      0.95      0.91        38

    accuracy                           0.89        61
   macro avg       0.89      0.86      0.87        61
weighted avg       0.89      0.89      0.88        61
```

K Nearest Neighbors

In [62]:
```
from sklearn.neighbors import KNeighborsClassifier
Kclassifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowsk
Kclassifier.fit(X_train, y_train)
```

Out[62]:
```
KNeighborsClassifier()
```

In [63]:
```
Y_pred = Kclassifier.predict(X_test)
```

In [64]:
```
from sklearn.metrics import confusion_matrix
Kcm = confusion_matrix(y_test, Y_pred)
```

In [65]:
```
Kcm
```

Out[65]:
```
array([[18,  5],
       [ 2, 36]])
```

In [66]:
```
print(classification_report(y_test,Y_pred))
```

```
              precision    recall  f1-score   support

           1       0.90      0.78      0.84        23
           3       0.88      0.95      0.91        38

    accuracy                           0.89        61
   macro avg       0.89      0.86      0.87        61
```

Naive Bayes

In [67]:
```
from sklearn.naive_bayes import GaussianNB
NBclassifier = GaussianNB()
NBclassifier.fit(X_train, y_train)
```

Out[67]:
```
GaussianNB()
```

In [68]:
```
Y_pred = classifier.predict(X_test)
```

In [69]:
```
from sklearn.metrics import confusion_matrix
NBcm = confusion_matrix(y_test, Y_pred)
```

In [70]:
```
NBcm
```

Out[70]:
```
array([[18,  5],
       [ 2, 36]])
```

In [71]:
```
print(classification_report(y_test,Y_pred))
```

```
              precision    recall  f1-score   support

           1       0.90      0.78      0.84        23
           3       0.88      0.95      0.91        38

    accuracy                           0.89        61
   macro avg       0.89      0.86      0.87        61
weighted avg       0.89      0.89      0.88        61
```