ICGC Data Preprocessing

In [2]:

```
#Upload Data Files
import pandas as pd
df = pd.read_csv('icgc_seq_annot.csv')
#print(df)

#Data Procesing - Remove hgnc_symbol that are NAs
df.dropna(subset=['hgnc_symbol'],inplace=True)
#print(df)

df.rename({'hgnc_symbol':'Patient_ID'}, axis=1, inplace=True)
df.drop('ensembl_gene_id', inplace=True, axis=1,errors='ignore')
print(df)
#Transpose
df = df.set_index('Patient_ID').T
#print('Number of resulting cols: ', len(df.columns))

#Raw data (Donor)
qc = pd.read_csv('donor.tsv', sep='\t')
#print(qc)
#print(len(qc))
#print(qc.head(1))
#print(qc.size)
#print(qc.shape)
qc.drop([0,0], axis=0)
#print(qc.iloc[:,[0,5]])

#Characterized Data (Donor)
qc.rename({'icgc_donor_id':'Patient_ID'}, axis=1, inplace=True)
qc.loc[qc['donor_vital_status'] == "deceased", 'donor_vital_status'] = '1'
qc.loc[qc['donor_vital_status'] == "alive", 'donor_vital_status'] = '0'
qc.rename({'donor_vital_status':'VS'}, axis=1, inplace=True)
qc1 = qc.iloc[:,[0,5]]
qc1

qc1.isnull().sum()
print(qc1.shape)
qc2=qc1.dropna(subset=['VS'])
print(qc2.shape)
qc2

#Merging Data
result=df.merge(qc2,right_on='Patient_ID', left_index=True)
result=result.set_index('Patient_ID')
result

#selecting columns
X= result.drop(['VS'],axis=1)
Y= result.iloc[:,[-1]]
Y = Y['VS']
```

```
      Patient_ID  DO32829  DO32860  DO32863  DO32875  DO32878  DO32887  \
0         TSPAN6     8.36    14.26    19.28    28.09    11.91     8.27
1           TNMD     0.00     0.03     0.04     0.02     0.00     0.00
2           DPM1    18.70    25.70    20.26    29.58    17.63     8.61
3          SCYL3     4.73     7.14     3.78     6.36     3.39     4.81
4       C1orf112     2.58     2.81     3.04     7.72     2.03     2.35
...          ...      ...      ...      ...      ...      ...      ...
49597     IPO5P1     1.05     0.79     0.53     0.48     0.52     2.51
49601     CCNYL6     0.00     0.00     0.00     0.00     0.00     0.00
49604     RNF225     0.07     0.04     0.06     0.08     0.17     0.14
49605       EGLN2    17.73    16.71    16.17    31.48    15.26    15.11
49611    VN1R79P     0.00     0.00     0.00     0.00     0.00     0.00

       DO32900  DO32936  DO33091  ...  DO49178  DO49181  DO49183  DO49184  \
0        22.89    28.34     3.04  ...     9.16     6.78    21.25     8.87
1         0.02     0.00     0.00  ...     0.38     0.11     0.00     0.00
2        20.02    60.79    61.48  ...    32.96    15.79    22.59     8.04
3         4.77     4.62     1.10  ...     6.69     7.46     9.57     9.81
4         3.57     5.42     3.78  ...     3.87     3.34     3.85     6.69
...        ...      ...      ...  ...      ...      ...      ...      ...
49597     3.54     0.57     1.05  ...     0.70     2.53     1.38     1.61
49601     0.00     0.00     0.01  ...     0.02     0.00     0.02     0.00
49604     0.04     0.14     0.00  ...     0.14     0.00     0.05     0.15
49605    17.97    18.04    15.80  ...    19.68    19.37    18.41    23.51
49611     0.00     0.00     0.00  ...     0.00     0.00     0.00     0.00

       DO49185  DO49193  DO49198  DO49199  DO49201  DO49204
0        12.53    12.68     3.75     5.60     2.85    13.73
1         0.00     0.00     0.00     0.00     0.00     0.00
2        25.03    21.69     4.95     7.31     9.06    39.19
3         7.59     7.41     5.23     4.16     7.41     2.94
4         4.63     3.50     2.01     2.13     3.70     5.43
...        ...      ...      ...      ...      ...      ...
49597     0.89     1.14     1.19     1.08     1.27     0.45
49601     0.00     0.00     0.00     0.00     0.00     0.00
49604     0.04     0.00     0.00     0.00     0.19     0.07
49605    14.03    14.89    27.91    16.74    16.72    20.39
49611     0.00     0.00     0.00     0.00     0.00     0.00

[35601 rows x 92 columns]
(461, 2)
(460, 2)
```

In [3]:
```
#Libraries
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC
from sklearn.datasets import make_moons
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import precision_score, recall_score, roc_auc_score, roc
from sklearn.model_selection import cross_val_predict

import pandas as pd
import matplotlib.pyplot as plt
import seaborn
from sklearn.impute import SimpleImputer
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC
from sklearn.datasets import make_moons
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score, roc_curve
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import accuracy_score
from sklearn.feature_selection import SelectKBest, f_classif

#For RFE
import sklearn
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestClassifier
import time
import datetime
```

In [4]:
```
#Correlation Test
cormat = result.corr()
round(cormat,2)
```

Out[4]:

	TSPAN6	TNMD	DPM1	SCYL3	C1orf112	FGR	CFH	FUCA2	GCLC	NFYA	...	Pl
TSPAN6	1.00	-0.19	0.47	0.25	0.12	-0.17	0.23	0.48	0.05	0.22	...	
TNMD	-0.19	1.00	-0.10	-0.03	0.56	0.06	-0.02	-0.14	-0.12	-0.20	...	
DPM1	0.47	-0.10	1.00	-0.04	0.11	-0.20	0.00	0.36	0.17	0.02	...	
SCYL3	0.25	-0.03	-0.04	1.00	0.04	0.18	0.19	0.29	-0.14	0.47	...	
C1orf112	0.12	0.56	0.11	0.04	1.00	-0.14	-0.12	-0.02	-0.01	-0.15	...	
...	...	...	...	...	...	...	...	...	...	...	...	
IPO5P1	-0.01	0.18	-0.17	0.13	0.32	-0.00	0.05	-0.15	-0.09	-0.09	...	
CCNYL6	-0.17	0.66	-0.21	-0.05	0.84	-0.16	-0.15	-0.22	-0.07	-0.25	...	
RNF225	0.03	-0.12	-0.03	-0.21	-0.10	0.08	-0.03	-0.18	-0.00	-0.13	...	
EGLN2	0.26	-0.18	0.14	0.06	-0.01	-0.09	0.14	0.38	0.26	0.13	...	
VN1R79P	0.08	-0.04	0.06	-0.07	-0.03	-0.10	-0.01	-0.05	-0.05	-0.02	...	

35601 rows × 35601 columns

In [ ]:
```
import seaborn as sns
sns.heatmap(cormat)
```

Feature Selection (RFE)

In-Built Function

In [5]:
```
#Full Function
def RFE_function(X,Y,learner,k,s,folds,score):
    #Start Timer
    t0= time.time()

    #RFE
    rfe=RFE(estimator=learner,n_features_to_select=k,step=s)
    select=rfe.fit(X,Y)

    #Result Matrix
    Xred = X.loc[:,rfe.support]

    #Stop Timer and Calculate Elapsed Time
    t1 = time.time() - t0
    t_s = t1
    t_hr = round(t_s/3600,3)
    return t_hr

    #Save Time as Txt File
    run_date = datetime.datetime.now()
    file_object = open("RFE_Info.txt","a")
    file_object.write('RFE | ' + str(estimator) + " | " + str(run_date) + " |
                      str(k) + ' | ' + str(t_hr) + "_hr")
    file_object.close()

    #Create CSV Name
    csv_name = str("ICGC_RFE_" + str(estimator) + "k_" + str(k)+ ".csv")
    print(csv_name)

    #Save to ININ4998 Pancreatic Cancer Folder
    display(Xred)
    Xred.to_csv(csv_name)
```

In [6]:
```
RFE_function(X,Y,RandomForestClassifier(),3,1,5,"accuracy")
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-6-f83abfdb1223> in <module>
----> 1 RFE_function(X,Y,RandomForestClassifier(),3,1,5,"accuracy")

<ipython-input-5-380c5f12e91f> in RFE_function(X, Y, learner, k, s, folds, sco
re)
      6     #RFE
      7     rfe=RFE(estimator=learner,n_features_to_select=k,step=s)
----> 8     select=rfe.fit(X,Y)
      9
     10     #Result Matrix

~/opt/anaconda3/lib/python3.8/site-packages/sklearn/feature_selection/_rfe.py
in fit(self, X, y)
    149             The target values.
    150         """
--> 151         return self._fit(X, y)
```

```
    152
    153     def _fit(self, X, y, step_score=None):

~/opt/anaconda3/lib/python3.8/site-packages/sklearn/feature_selection/_rfe.py
in _fit(self, X, y, step_score)
    200                 coefs = estimator.coef_
    201             else:
--> 202                 coefs = getattr(estimator, 'feature_importances_', Non
e)
    203             if coefs is None:
    204                 raise RuntimeError('The classifier does not expose '

~/opt/anaconda3/lib/python3.8/site-packages/sklearn/ensemble/_forest.py in fea
ture_importances_(self)
    452             return np.zeros(self.n_features_, dtype=np.float64)
    453
--> 454         all_importances = np.mean(all_importances,
    455                                   axis=0, dtype=np.float64)
    456         return all_importances / np.sum(all_importances)

<__array_function__ internals> in mean(*args, **kwargs)

~/opt/anaconda3/lib/python3.8/site-packages/numpy/core/fromnumeric.py in mean(
a, axis, dtype, out, keepdims)
   3370             return mean(axis=axis, dtype=dtype, out=out, **kwargs)
   3371
-> 3372     return _methods._mean(a, axis=axis, dtype=dtype,
   3373                           out=out, **kwargs)
   3374

~/opt/anaconda3/lib/python3.8/site-packages/numpy/core/_methods.py in _mean(a,
axis, dtype, out, keepdims)
    142
    143 def _mean(a, axis=None, dtype=None, out=None, keepdims=False):
--> 144     arr = asanyarray(a)
    145
    146     is_float16_result = False

~/opt/anaconda3/lib/python3.8/site-packages/numpy/core/_asarray.py in asanyarr
ay(a, dtype, order)
    134
    135     """
--> 136     return array(a, dtype, copy=False, order=order, subok=True)
    137
    138

KeyboardInterrupt:
```

Separate Coding

In [ ]:

```
#RFE Parameters
estimator = RandomForestClassifier()
```

In [ ]:
```
#Start Timer
t0= time.time()

#RFE
rfe=RFE(estimator,n_features_to_select=20)
select=rfe.fit(X,Y)

#Stop Timer and Calculate Elapsed Time
t1 = time.time() - t0

#Calculate Elapsed Time
print("Time elapsed (seconds): ", t1) # CPU seconds elapsed (floating point)
print("Time elapsed (hours):", t1/3600)
```

In [ ]:
```
#Select headings for the X values
#forward = ["SHROOM3", "TMEM187", "TRIM29"]
Xred = X.loc[:,select]
#Xred = X.loc[:,forward]
```

In [ ]:
```
#Create CSV Name
k_features = len(Xred.columns)
csv_name = str("RFE_" + str(estimator) + "k_" + str(k_features)+ ".csv")
print(csv_name)

#Save to ININ4998 Pancreatic Cancer Folder
display(Xred)
Xred.to_csv(csv_name)
```

In [ ]:
```
#Save Time as Txt File
import datetime

run_date = datetime.datetime.now()
file_object = open("RFE_Info.txt","a")
file_object.write('RFE | ' + str(estimator) + " | " + str(run_date) + " | " +
                  str(k_features) + ' | ' + str(t1) + "_hr")
file_object.close()
```

Classification Models

In [ ]:
```
#Extracting CSV File From Feature Selection (Manual Input)
X_csv = pd.read_csv(csv_name) #Example: 'RFE_RandomForestClassifier()k_3.csv'
X_FS = X_csv.set_index("PatientID")
display(X_FS)
```

In [ ]:
```
#Model 1: Logistic Regression
def createLogisticRegression():
    logreg= Pipeline([
        ("scaler", StandardScaler()),
        ("logistic_reg", LogisticRegression(max_iter=500)),
    ])
    return logreg

#Model 2: SVC
from sklearn.ensemble import RandomForestClassifier
def createSVC():
    pipe= Pipeline([
        ("scaler", StandardScaler()),
        ("classifier", SVC(decision_function_shape="ovo")),
    ])
    return pipe

#Model 3: Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
def createRandomForestClassifier():
    pipe= Pipeline([
        ("scaler", StandardScaler()),
        ("classifier", RandomForestClassifier(n_estimators=100, max_depth=3,
    ])
    return pipe
```

In [ ]:
```
#Saving ROC Curves
def predict_with_data(data_x, data_y, classifier, figname=''):
    classifier.fit(data_x,data_y)
    y_scores = cross_val_predict(classifier, data_x, data_y, cv=10)
    scores = cross_val_score(classifier, data_x, data_y, scoring='accuracy',

    #graph ROC curve
    fpr, tpr, thresholds= roc_curve(data_y,y_scores)
    #save_plot_roc_curve(fpr, tpr, figname)

    return (precision_score(data_y, y_scores),recall_score(data_y, y_scores),
            roc_auc_score(data_y,y_scores), accuracy_score(data_y,y_scores))
```

In [ ]:
```
#from sklearn.preprocessing import OrdinalEncoder
#ord_enc = OrdinalEncoder()
#code = ord_enc.fit_transform(code)
#Y.values
numres=pd.get_dummies(Y)
display(numres)
```

In [ ]:
```
#Logistic Regression Score
precision_log,recall_log,roc_log,acu_log=predict_with_data(Xred,numres.loc[:,

print('Logistic Regression')
print('---------')
print('Precision: ', precision_log)
print('Recall: ', recall_log)
print('ROC AUC Score: ', roc_log)
print ('Accuracy: ', acu_log)
print('---------')
```

In [ ]:
```
#SVC Score
precision_svc,recall_svc,roc_svc,acu_svc=predict_with_data(Xred,numres.loc[:,

print('SVC')
print('---------')
print('Precision: ', precision_svc)
print('Recall: ', recall_svc)
print('ROC AUC Score: ', roc_svc)
print ('Accuracy: ', acu_svc)
print('---------')
```

In [ ]:
```
#Random Forest Classifier Score
precision_rf,recall_rf,roc_rf,acu_rf=predict_with_data(Xred,numres.loc[:,'DEC
                                                       createRandomForestClas

print('RandomForest')
print('---------')
print('Precision: ', precision_rf)
print('Recall: ', recall_rf)
print('ROC AUC Score: ', roc_rf)
print ('Accuracy: ', acu_rf)
print('---------')
```