TCGA Data Preprocessing

In [1]:

```
#Importing data set 1
import pandas as pd
df = pd.read_csv("data_clinical_patient1.txt", sep='\t')
#print(df)
#print (len(df))
#print (df.head(1))
#print (df.size)
#print (df.shape)

#counting Na's in data set 1
df.isnull()
df.isnull().sum().sum()
df.isnull().sum()

#characterizing data
df.loc[df['Overall Survival Status']== "1:DECEASED", 'Overall Survival Status
df.loc[df['Overall Survival Status']== "0:LIVING", 'Overall Survival Status']
df.drop([0,1,2,3],axis=0,inplace=True)

#changing column name
df.rename(columns={"#Patient Identifier": "PatientID"},inplace=True)
df.rename(columns={"Overall Survival Status": "OS"},inplace=True)
df

#selecting columns to analyze
nr= df.iloc[:, [0,30]] #columnas a analizar
nr

#importing Data Set 2
df2 = pd.read_csv("data_RNA_Seq_v2_mRNA_median_Zscores1.txt", sep='\t')
#print (len(df2))
#print (df2.head(1))
#print (df2.size)
#print (df2.shape)

df2.drop('Entrez_Gene_Id', inplace=True,axis=1)
df2

#changing column name & remove NAs
print(df2.shape)
df2.dropna(subset=['Hugo_Symbol'],inplace=True)
df2.rename(columns={"Hugo_Symbol": "PatientID"},inplace=True)
print(df2.shape)
df2

#transposing data set
df3 = df2.set_index('PatientID').T
df3
```

```
#pre-processing data set
df3.index=df3.index.map(lambda x:str(x)[:-3])
df3.index

#Counting Na's in columns & estimating how many genes with complete column of
df3.isnull().sum().sum()/177

#eliminating Na's
#print(df3.shape)
dfn=df3.dropna(1)
#print(dfn.shape)
dfn

#merge data sets
result=dfn.merge(nr,right_on='PatientID', left_index=True)
result=result.set_index('PatientID')
result

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

#finding data dimensions
#print(X.shape)
#print(Y.shape)

Y = Y['OS']
```

```
(20531, 178)
(20502, 178)
```

In [2]:

```
display(result)
```

	A1BG	A1CF	A2BP1	A2LD1	A2M	A2ML1	A4GALT	A4GNT	AAA1	
PatientID										
TCGA-2J-AAB1	-0.2628	-0.4960	-0.2204	0.6514	0.3467	-0.1833	1.4500	2.0312	-0.0209	-0
TCGA-2J-AAB4	-0.4560	-0.1130	-0.0204	-0.2575	-0.0987	-0.2662	0.8009	-0.2648	-0.5009	-0
TCGA-2J-AAB6	-0.2579	-0.7303	-0.2807	0.4239	-0.9420	1.4869	0.7441	-0.5393	-0.2661	0
TCGA-2J-AAB8	-0.4546	-0.3804	-0.2807	0.7210	-0.2627	-0.2498	-0.4786	-0.4906	-0.3381	0
TCGA-2J-AAB9	-0.0794	-0.5496	-0.1481	0.4708	0.8775	0.3537	0.7514	-0.2482	-0.6330	0
...	...	...	...	...	...	...	...	...	...	
TCGA-XN-A8T5	0.0515	-0.7119	0.2353	-0.5462	0.4042	-0.2546	-0.2218	-0.2253	-0.6330	-1
TCGA-YB-A89D	0.3130	0.1672	-0.1536	-0.3490	0.7868	-0.2592	0.3078	-0.4276	0.4954	-0
TCGA-YH-A8SY	0.9465	-0.8101	-0.2807	-0.5668	-0.9510	1.1341	1.8806	-0.5358	-0.6330	-0
TCGA-YY-A8LH	-0.3863	-0.4330	0.2102	-0.1282	-1.4081	-0.2576	-0.6245	0.0408	0.4736	0
TCGA-Z5-AAPL	0.0364	-0.7876	-0.2807	-0.3138	-0.7489	-0.2320	-0.2572	-0.5192	-0.6330	0

177 rows × 20026 columns

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 [5]:

```
#Correlation Test
cormat = result.corr()
round(cormat,2)
```

```
Out[5]:
             A1BG   A1CF  A2BP1  A2LD1    A2M  A2ML1  A4GALT  A4GNT   AAA1   AAS  ...  ZWIL
     A1BG    1.00   0.14  -0.07  -0.14   0.03  -0.07    0.02  -0.07   0.07  0.53  ...   -0.
     A1CF    0.14   1.00   0.14  -0.04  -0.06  -0.17   -0.29   0.07   0.16  0.22  ...   -0.
    A2BP1   -0.07   0.14   1.00  -0.04  -0.09  -0.04   -0.07  -0.03  -0.08  0.15  ...    0.
    A2LD1   -0.14  -0.04  -0.04   1.00  -0.05   0.06   -0.08   0.10   0.01 -0.05  ...   -0.
      A2M    0.03  -0.06  -0.09  -0.05   1.00  -0.16   -0.04   0.03  -0.12 -0.26  ...   -0.
      ...     ...    ...    ...    ...    ...    ...     ...    ...    ...   ...  ...
   ZYG11A   -0.00  -0.07   0.22  -0.09  -0.03  -0.03    0.09  -0.08  -0.15  0.17  ...    0.
   ZYG11B   -0.00   0.10  -0.05  -0.24   0.54   0.03   -0.18  -0.06  -0.05 -0.40  ...    0.
      ZYX   -0.18  -0.41  -0.14   0.03   0.12   0.07    0.37  -0.11  -0.07  0.00  ...    0.
    ZZEF1    0.38   0.53   0.10  -0.14   0.22  -0.17   -0.23   0.03   0.09  0.23  ...   -0.
     ZZZ3   -0.08   0.27   0.07  -0.27   0.21   0.12   -0.22   0.03   0.04 -0.22  ...    0.

20025 rows × 20025 columns
```

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

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-7-cf75107f6731> in <module>
      1 import seaborn as sns
----> 2 sns.heatmap(cormat)

~/opt/anaconda3/lib/python3.8/site-packages/seaborn/_decorators.py in inner_f(
*args, **kwargs)
     44             )
     45         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)}
)
---> 46         return f(**kwargs)
     47     return inner_f
     48

~/opt/anaconda3/lib/python3.8/site-packages/seaborn/matrix.py in heatmap(data,
vmin, vmax, cmap, center, robust, annot, fmt, annot_kws, linewidths, linecolor
, cbar, cbar_kws, cbar_ax, square, xticklabels, yticklabels, mask, ax, **kwarg
s)
    556     if square:
    557         ax.set_aspect("equal")
--> 558     plotter.plot(ax, cbar_ax, kwargs)
    559     return ax
    560

~/opt/anaconda3/lib/python3.8/site-packages/seaborn/matrix.py in plot(self, ax
, cax, kws)
```

```
    340         # Possibly rotate them if they overlap
    341         if hasattr(ax.figure.canvas, "get_renderer"):
--> 342             ax.figure.draw(ax.figure.canvas.get_renderer())
    343         if axis_ticklabels_overlap(xtl):
    344             plt.setp(xtl, rotation="vertical")

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/artist.py in draw_wrapper(artist, renderer, *args, **kwargs)
     39                 renderer.start_filter()
     40 
---> 41             return draw(artist, renderer, *args, **kwargs)
     42         finally:
     43             if artist.get_agg_filter() is not None:

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/figure.py in draw(self, renderer)
   1861 
   1862             self.patch.draw(renderer)
-> 1863             mimage._draw_list_compositing_images(
   1864                 renderer, self, artists, self.suppressComposite)
   1865 

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/image.py in _draw_list_compositing_images(renderer, parent, artists, suppress_composite)
    129     if not_composite or not has_images:
    130         for a in artists:
--> 131             a.draw(renderer)
    132     else:
    133         # Composite any adjacent images together

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/artist.py in draw_wrapper(artist, renderer, *args, **kwargs)
     39                 renderer.start_filter()
     40 
---> 41             return draw(artist, renderer, *args, **kwargs)
     42         finally:
     43             if artist.get_agg_filter() is not None:

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/cbook/deprecation.py in wrapper(*inner_args, **inner_kwargs)
    409                          else deprecation_addendum,
    410                 **kwargs)
--> 411         return func(*inner_args, **inner_kwargs)
    412 
    413     return wrapper

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/axes/_base.py in draw(self, renderer, inframe)
   2745             renderer.stop_rasterizing()
   2746 
-> 2747         mimage._draw_list_compositing_images(renderer, self, artists)
   2748 
   2749         renderer.close_group('axes')

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/image.py in _draw_list_compositing_images(renderer, parent, artists, suppress_composite)
    129     if not_composite or not has_images:
```

```
    130         for a in artists:
--> 131             a.draw(renderer)
    132     else:
    133         # Composite any adjacent images together

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/artist.py in draw_wrapp
er(artist, renderer, *args, **kwargs)
     39                 renderer.start_filter()
     40 
---> 41             return draw(artist, renderer, *args, **kwargs)
     42         finally:
     43             if artist.get_agg_filter() is not None:

~/opt/anaconda3/lib/python3.8/site-packages/matplotlib/collections.py in draw(
self, renderer)
   2046                 gc, triangles, colors, transform.frozen())
   2047         else:
-> 2048             renderer.draw_quad_mesh(
   2049                 gc, transform.frozen(), self._meshWidth, self._meshHei
ght,
   2050                 coordinates, offsets, transOffset,

KeyboardInterrupt: 
```

Feature Selection (RFE)

In-Built Function

In [7]:

```
#Full Function
def RFE_function(estimator,k):
    #Start Timer
    t0= time.time()

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

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

    #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("TCGA_RFE_" + str(estimator) + "k_" + str(k)+ ".csv")
    print(csv_name)

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

Separate Coding

In [5]:

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

In [4]:
```
#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)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-4-84b039f769e4> in <module>
      4 #RFE
      5 rfe=RFE(estimator,n_features_to_select=20)
----> 6 select=rfe.fit(X,Y)
      7 
      8 #Stop Timer and Calculate Elapsed Time

~/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)
    194                 print("Fitting estimator with %d features." % np.sum(s
upport_))
    195 
--> 196             estimator.fit(X[:, features], y)
    197 
    198             # Get coefs

~/opt/anaconda3/lib/python3.8/site-packages/sklearn/ensemble/_forest.py in fit
(self, X, y, sample_weight)
    301                 "sparse multilabel-indicator for y is not supported."
    302             )
--> 303         X, y = self._validate_data(X, y, multi_output=True,
    304                                    accept_sparse="csc", dtype=DTYPE)
    305         if sample_weight is not None:

~/opt/anaconda3/lib/python3.8/site-packages/sklearn/base.py in _validate_data(
self, X, y, reset, validate_separately, **check_params)
    430                 y = check_array(y, **check_y_params)
    431             else:
--> 432                 X, y = check_X_y(X, y, **check_params)
    433             out = X, y
    434 
```

```
~/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py in inn
er_f(*args, **kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)}
)
---> 72         return f(**kwargs)
     73     return inner_f
     74 

~/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py in che
ck_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order, copy, force_all
_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ensure_min_fea
tures, y_numeric, estimator)
    793         raise ValueError("y cannot be None")
    794 
--> 795     X = check_array(X, accept_sparse=accept_sparse,
    796                     accept_large_sparse=accept_large_sparse,
    797                     dtype=dtype, order=order, copy=copy,

~/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py in inn
er_f(*args, **kwargs)
     70                           FutureWarning)
     71         kwargs.update({k: arg for k, arg in zip(sig.parameters, args)}
)
---> 72         return f(**kwargs)
     73     return inner_f
     74 

~/opt/anaconda3/lib/python3.8/site-packages/sklearn/utils/validation.py in che
ck_array(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_
all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, esti
mator)
    596                     array = array.astype(dtype, casting="unsafe", copy
=False)
    597                 else:
--> 598                     array = np.asarray(array, order=order, dtype=dtype
)
    599             except ComplexWarning:
    600                 raise ValueError("Complex data not supported\n"

~/opt/anaconda3/lib/python3.8/site-packages/numpy/core/_asarray.py in asarray(
a, dtype, order)
     81 
     82     """
---> 83     return array(a, dtype, copy=False, order=order)
     84 
     85 

KeyboardInterrupt: 
```

In [5]:

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

In [6]:

```
#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)
```

```
RFE_RandomForestClassifier()k_3.csv
```

	SHROOM3	TMEM187	TRIM29
PatientID			
TCGA-2J-AAB1	2.2840	0.4545	1.4970
TCGA-2J-AAB4	0.5049	1.0436	1.4909
TCGA-2J-AAB6	-1.1850	0.9344	1.1323
TCGA-2J-AAB8	-0.4312	-1.0370	0.1052
TCGA-2J-AAB9	-0.5645	0.1825	0.4217
...	...	...	...
TCGA-XN-A8T5	-1.1185	0.2566	-0.4477
TCGA-YB-A89D	-0.4086	-0.9106	0.4496
TCGA-YH-A8SY	-0.5280	-0.1551	0.4698
TCGA-YY-A8LH	0.2977	1.3603	-0.3586
TCGA-Z5-AAPL	-1.5322	-1.3257	-0.4296

177 rows × 3 columns

In [10]:

```
#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 [24]:

```python
#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)
```

	SHROOM3	TMEM187	TRIM29
PatientID			
TCGA-2J-AAB1	2.2840	0.4545	1.4970
TCGA-2J-AAB4	0.5049	1.0436	1.4909
TCGA-2J-AAB6	-1.1850	0.9344	1.1323
TCGA-2J-AAB8	-0.4312	-1.0370	0.1052
TCGA-2J-AAB9	-0.5645	0.1825	0.4217
...	...	...	...
TCGA-XN-A8T5	-1.1185	0.2566	-0.4477
TCGA-YB-A89D	-0.4086	-0.9106	0.4496
TCGA-YH-A8SY	-0.5280	-0.1551	0.4698
TCGA-YY-A8LH	0.2977	1.3603	-0.3586
TCGA-Z5-AAPL	-1.5322	-1.3257	-0.4296

177 rows × 3 columns

In [15]:

```
#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 [16]:

```
#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 [17]:

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

	DECEASED	LIVING
PatientID		
TCGA-2J-AAB1	1	0
TCGA-2J-AAB4	0	1
TCGA-2J-AAB6	1	0
TCGA-2J-AAB8	0	1
TCGA-2J-AAB9	1	0
...	...	...
TCGA-XN-A8T5	0	1
TCGA-YB-A89D	0	1
TCGA-YH-A8SY	0	1
TCGA-YY-A8LH	0	1
TCGA-Z5-AAPL	0	1

177 rows × 2 columns

In [19]:

```
#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('----------')
```

```
Logistic Regression
---------
Precision:  0.6086956521739131
Recall:  0.6086956521739131
ROC AUC Score:  0.5925831202046036
Accuracy:  0.5932203389830508
---------
```

In [25]:

```
#Logistic Regression Score
precision_log,recall_log,roc_log,acu_log=predict_with_data(X_FS,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('---------')
```

```
Logistic Regression
---------
Precision:  0.6086956521739131
Recall:  0.6086956521739131
ROC AUC Score:  0.5925831202046036
Accuracy:  0.5932203389830508
---------
```

In [18]:

```
#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('---------')
```

```
SVC
---------
Precision:  0.6283185840707964
Recall:  0.7717391304347826
ROC AUC Score:  0.6388107416879796
Accuracy:  0.6440677966101694
---------
```

In [19]:

```
#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('---------')
```

```
RandomForest
---------
Precision:  0.6363636363636364
Recall:  0.7608695652173914
ROC AUC Score:  0.6451406649616368
Accuracy:  0.6497175141242938
---------
```

```
In [ ]:
```