Machine learning predicts metastatic progression using novel differentially expressed lncRNAs as potential markers in pancreatic cancer

Hasan Alsharoh

In [1]:
```
import glob
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from scipy.stats import ttest_ind
import os
from pydeseq2.dds import DeseqDataSet
from pydeseq2.ds import DeseqStats
from sanbomics.plots import volcano
import gseapy as gp
from gseapy import barplot, dotplot
from gseapy.plot import gseaplot
from sanbomics.tools import id_map
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_curve, auc, precision_recall_curve
from sklearn.svm import SVC
from xgboost.sklearn import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
import xgboost as xgb
from skopt import BayesSearchCV
```

ML models

First import data

In [2]:
```
sigml = pd.read_csv(r"\ml.txt", sep = '\t') #load ml file with data
sigml= sigml.drop(columns= ['Unnamed: 0'])
```

In [3]:
```
smote = SMOTE() # Use smote to reduce bias
```

Defining X, y, for training ML models

In [45]:
```
X = sigml.iloc[:,:-1]
y = sigml['state']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state
```

In [46]:
```
features = list(X.columns)
```

In [47]:
```
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
```

Logistic Regression

In [48]:
```
lrmodel= LogisticRegression(C=10.0, class_weight='balanced', max_iter=10000, multi_cla
lrmodel.fit(X_resampled, y_resampled)
```

Out[48]:
```
LogisticRegression(C=10.0, class_weight='balanced', max_iter=10000,
                   multi_class='ovr')
```

In [49]:
```
y_pred = lrmodel.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Confusion Matrix:")
print(cm)
```

```
Accuracy: 0.7391304347826086
Precision: 0.7631578947368421
Recall: 0.90625
F1 Score: 0.8285714285714286
Confusion Matrix:
[[ 5  9]
 [ 3 29]]
```

In [50]:
```
report = classification_report(y_test,y_pred) # for a report regarding the evaluation
print(report)
```

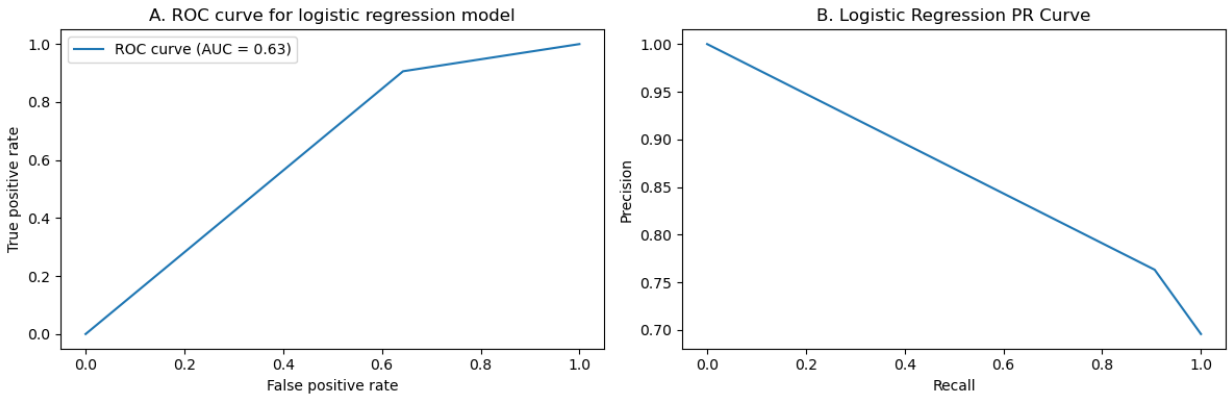
```
              precision    recall  f1-score   support

           0       0.62      0.36      0.45        14
           1       0.76      0.91      0.83        32

    accuracy                           0.74        46
   macro avg       0.69      0.63      0.64        46
weighted avg       0.72      0.74      0.71        46
```

Coefficients

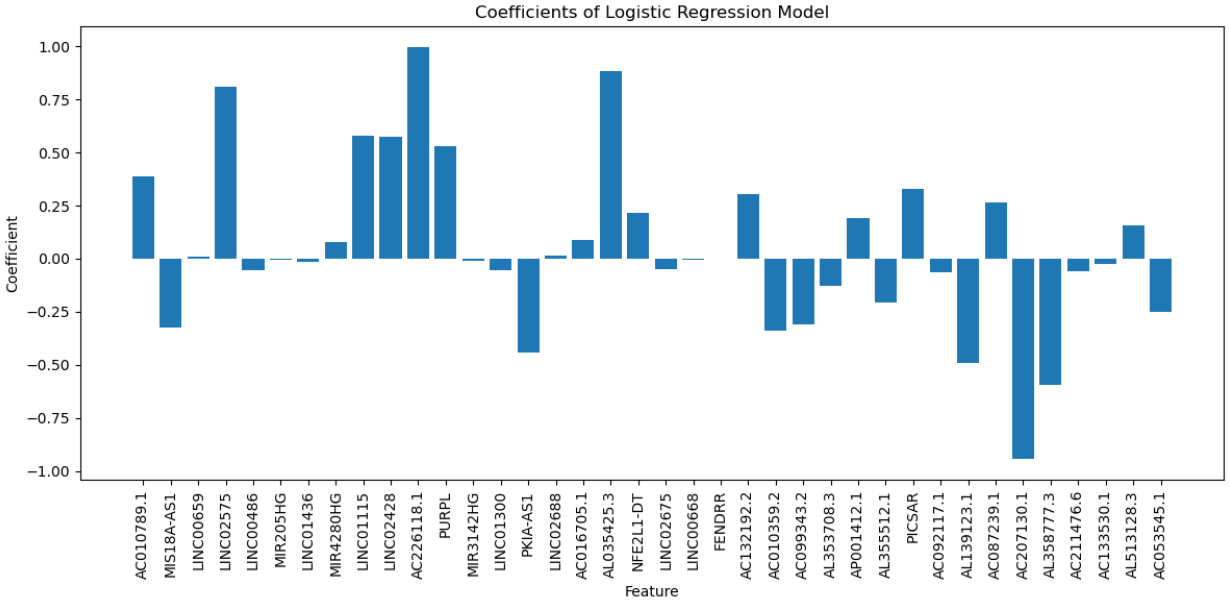
In [51]:
```
lrcoefs = lrmodel.coef_ # take coefficients list
sortedlrcoefsarr = sorted(lrcoefs, key= abs) # sort the coefficients
sortedlrcoefs = [x for y in sortedlrcoefsarr for x in y] # make a list of the list in
```

Figures LR

In [52]:
```
fpr, tpr, thresholds = roc_curve(y_test, y_pred) # to retrieve false positive and true
auc = metrics.auc(fpr,tpr) # calculate AUC
precision, recall, thresholds = precision_recall_curve(y_test, y_pred) # calculate PR
# Create a new figure with two subplots side by side
plt.figure(figsize=(12, 4))

# First subplot for ROC curve
plt.subplot(121)  # 1 row, 2 columns, first subplot
plt.plot(fpr, tpr, label='ROC curve (AUC = {:.2f})'.format(auc))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('A. ROC curve for logistic regression model')
plt.legend()

# Second subplot for PR curve
plt.subplot(122)  # 1 row, 2 columns, second subplot
plt.plot(recall, precision)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('B. Logistic Regression PR Curve')

plt.tight_layout()  # So that the subplots don't overlap

plt.show()
```



In [53]:
```
plt.figure(figsize = (12,6))
plt.bar(features,sortedlrcoefs)
plt.xticks(rotation = 90)
plt.xlabel("Feature")
plt.ylabel("Coefficient")
plt.title("Coefficients of Logistic Regression Model")
plt.tight_layout()
plt.show()
```



Support Vector Machine

In [54]:
```
svmbest= SVC(C=100,class_weight= 'balanced', coef0=0.05, degree=2, gamma='auto', kerne
```

In [55]:
```
svmbest.fit(X_resampled,y_resampled)
```

Out[55]:
```
SVC(C=100, class_weight='balanced', coef0=0.05, degree=2, gamma='auto',
    kernel='poly')
```

In [56]:
```
y_pred = svmbest.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Confusion Matrix:")
print(cm)
```

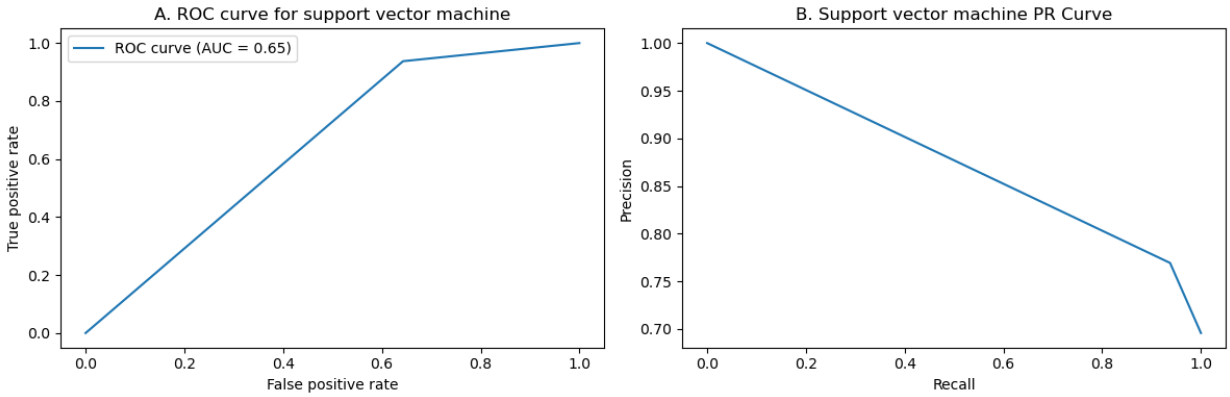
```
Accuracy: 0.7608695652173914
Precision: 0.7692307692307693
Recall: 0.9375
F1 Score: 0.8450704225352113
Confusion Matrix:
[[ 5  9]
 [ 2 30]]
```

In [57]:
```
report = classification_report(y_test,y_pred)
print(report)
```

	precision	recall	f1-score	support
0	0.71	0.36	0.48	14
1	0.77	0.94	0.85	32
accuracy			0.76	46
macro avg	0.74	0.65	0.66	46
weighted avg	0.75	0.76	0.73	46

SVM figures

In [58]:
```
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
auc = metrics.auc(fpr,tpr)
precision, recall, thresholds = precision_recall_curve(y_test, y_pred)

plt.figure(figsize=(12, 4))

# First subplot for ROC curve
plt.subplot(121)  # 1 row, 2 columns, first subplot
plt.plot(fpr, tpr, label='ROC curve (AUC = {:.2f})'.format(auc))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('A. ROC curve for support vector machine')
plt.legend()

# Second subplot for PR curve
plt.subplot(122)  # 1 row, 2 columns, second subplot
plt.plot(recall, precision)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('B. Support vector machine PR Curve')

plt.tight_layout()  # So that the subplots don't overlap

plt.show()
```



Random Forest Classifier

In [59]:
```
bestfc = RandomForestClassifier(ccp_alpha=0.0,
                                class_weight= 'balanced',
                                max_depth=9,
                                max_features='log2',
                               max_leaf_nodes=18,
```

```
                                max_samples=16,
                                min_impurity_decrease= 0.05,
                                min_samples_split=3,
                                 min_samples_leaf=1,
                                min_weight_fraction_leaf=0.07,
                                n_estimators=493,
                                criterion = 'entropy')
```

In [65]:
```
bestfc.fit(X_resampled,y_resampled)
```

Out[65]:
```
RandomForestClassifier(class_weight='balanced', criterion='entropy',
                       max_depth=9, max_features='log2', max_leaf_nodes=18,
                       max_samples=16, min_impurity_decrease=0.05,
                       min_samples_split=3, min_weight_fraction_leaf=0.07,
                       n_estimators=493)
```

In [79]:
```
y_pred= bestfc.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Confusion Matrix:")
print(cm)
```

```
Accuracy: 0.7608695652173914
Precision: 0.8620689655172413
Recall: 0.78125
F1 Score: 0.8196721311475409
Confusion Matrix:
[[10  4]
 [ 7 25]]
```

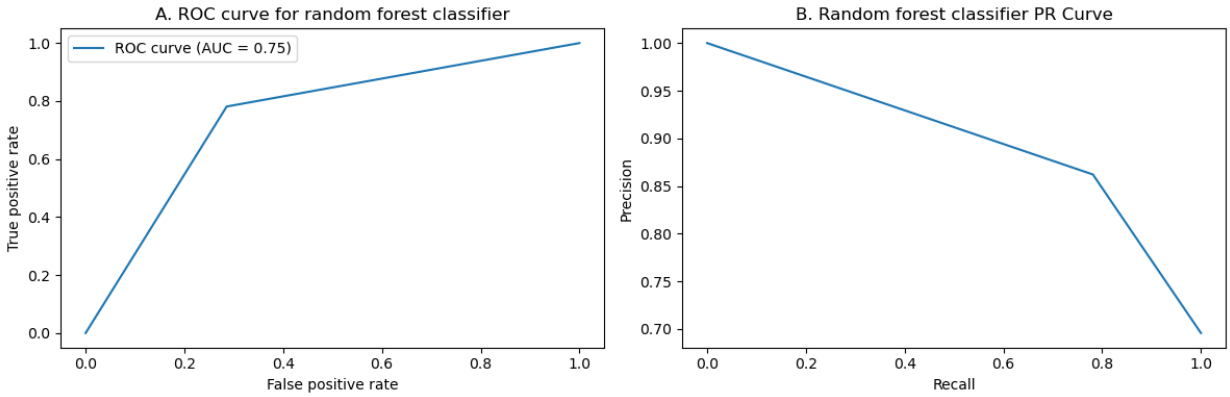
In [80]:
```
report = classification_report(y_test,y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.59      0.71      0.65        14
           1       0.86      0.78      0.82        32

    accuracy                           0.76        46
   macro avg       0.73      0.75      0.73        46
weighted avg       0.78      0.76      0.77        46
```

Figures

In [67]:
```
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
auc = metrics.auc(fpr,tpr)
precision, recall, thresholds = precision_recall_curve(y_test, y_pred)
```

```
plt.figure(figsize=(12, 4))
# previously defined AUC and fpr/tpr for single figures
# First subplot for ROC curve
plt.subplot(121)  # 1 row, 2 columns, first subplot
plt.plot(fpr, tpr, label='ROC curve (AUC = {:.2f})'.format(auc))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('A. ROC curve for random forest classifier')
plt.legend()

# Second subplot for PR curve
plt.subplot(122)  # 1 row, 2 columns, second subplot
plt.plot(recall, precision)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('B. Random forest classifier PR Curve')

plt.tight_layout()  # So that the subplots don't overlap

plt.show()
```



XGBoost Classifier

In [69]:
```
goodx = xgb.XGBClassifier(base_score= 0.16,
                          booster= 'dart',
                          colsample_bylevel= 0.2,
                          colsample_bynode=0.2,
                          colsample_bytree= 0.2,
                          gamma= 0.15,
                          grow_policy= 'lossguide',
                          importance_type= 'cover',
                          learning_rate= 0.0925,
                          max_bin=38,
                          max_delta_step= 12.05,
                          max_depth=10,
                          max_leaves=21,
                          min_child_weight= 0.2,
                          n_estimators= 230,
                          num_parallel_tree= 7,
                          objective= 'binary:logistic',
                          reg_lambda= 0.02,
                          subsample=0.85
                         )
```

In [75]:
```
goodx.fit(X_resampled,y_resampled)
y_pred = goodx.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Confusion Matrix:")
print(cm)
```

```
Accuracy: 0.717391304347826
Precision: 0.7317073170731707
Recall: 0.9375
F1 Score: 0.821917808219178
Confusion Matrix:
[[ 3 11]
 [ 2 30]]
```

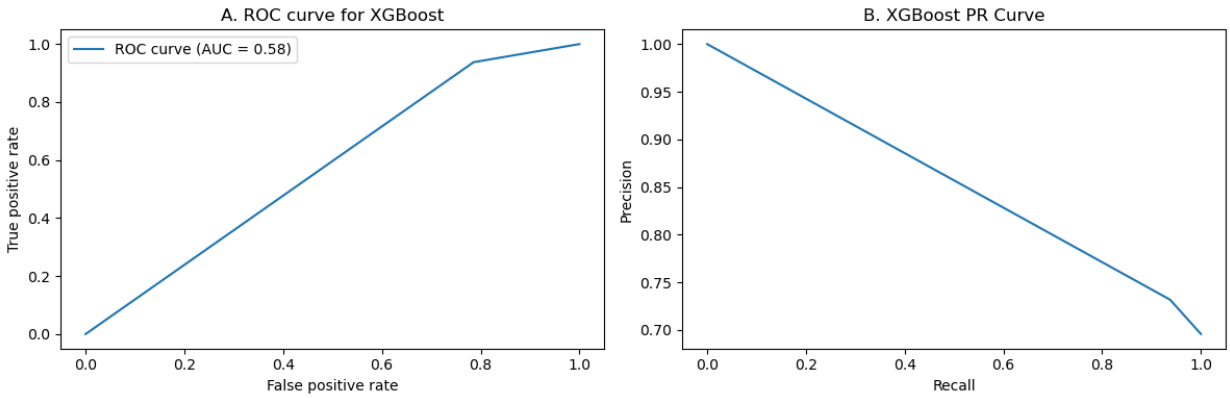
In [77]:
```
report = classification_report(y_test,y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.60      0.21      0.32        14
           1       0.73      0.94      0.82        32

    accuracy                           0.72        46
   macro avg       0.67      0.58      0.57        46
weighted avg       0.69      0.72      0.67        46
```

Figure

In [78]:
```
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
auc = metrics.auc(fpr,tpr)
precision, recall, thresholds = precision_recall_curve(y_test, y_pred)

plt.figure(figsize=(12, 4))
# previously defined AUC and fpr/tpr for single figures
# First subplot for ROC curve
plt.subplot(121)  # 1 row, 2 columns, first subplot
plt.plot(fpr, tpr, label='ROC curve (AUC = {:.2f})'.format(auc))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('A. ROC curve for XGBoost')
plt.legend()

# Second subplot for PR curve
plt.subplot(122)  # 1 row, 2 columns, second subplot
plt.plot(recall, precision)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('B. XGBoost PR Curve')

plt.tight_layout()  # So that the subplots don't overlap
```

```
plt.show()
```


A. ROC curve for XGBoost
ROC curve (AUC = 0.58)
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0
B. XGBoost PR Curve
Precision
Recall
1.00
0.95
0.90
0.85
0.80
0.75
0.70