In [139]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [140]:
```
data = pd.read_csv('E:/New folder/data_banknote_authentication.txt', header=None)
data.head()
```

Out[140]:

	0	1	2	3	4
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

In [141]:
```
data.shape
```

Out[141]:
```
(1372, 5)
```

In [142]:
```
data.columns
```

Out[142]:
```
Int64Index([0, 1, 2, 3, 4], dtype='int64')
```

In [143]:
```
data.columns = ['Variance', 'Skewness', 'Kurtosis', 'Entropy', 'Class']
data.head()
```

Out[143]:

	Variance	Skewness	Kurtosis	Entropy	Class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

In [144]:
```
data.Class.value_counts()
```

Out[144]:
```
0    762
1    610
Name: Class, dtype: int64
```

In [145]:
```
data.Variance.isna().sum()
```

Out[145]:
```
0
```

In [146]:
```
data.Skewness.isna().sum()
```

Out[146]:
```
0
```

In [147]:
```
data.Kurtosis.isna().sum()
```

Out[147]:
```
0
```

In [148]:
```
data.Entropy.isna().sum()
```

Out[148]:
```
0
```

In [149]:
```
data.head(10)
```

Out[149]:

	Variance	Skewness	Kurtosis	Entropy	Class
0	3.62160	8.6661	-2.80730	-0.44699	0
1	4.54590	8.1674	-2.45860	-1.46210	0
2	3.86600	-2.6383	1.92420	0.10645	0
3	3.45660	9.5228	-4.01120	-3.59440	0
4	0.32924	-4.4552	4.57180	-0.98880	0
5	4.36840	9.6718	-3.96060	-3.16250	0
6	3.59120	3.0129	0.72888	0.56421	0
7	2.09220	-6.8100	8.46360	-0.60216	0
8	3.20320	5.7588	-0.75345	-0.61251	0
9	1.53560	9.1772	-2.27180	-0.73535	0

In [150]:
```
data.info()
```

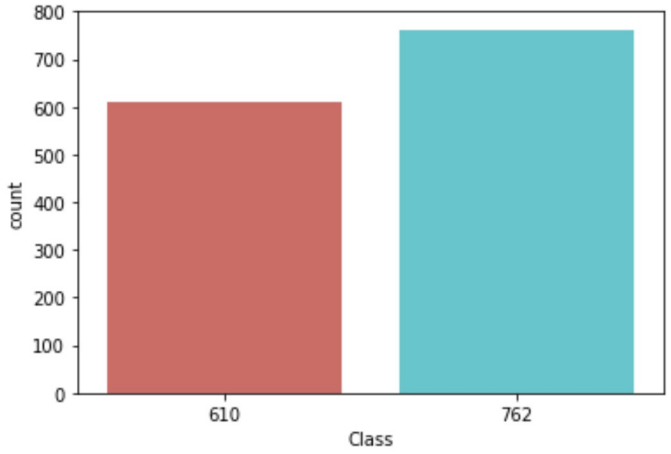
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1372 entries, 0 to 1371
Data columns (total 5 columns):
Variance    1372 non-null float64
Skewness    1372 non-null float64
Kurtosis    1372 non-null float64
Entropy     1372 non-null float64
Class       1372 non-null int64
dtypes: float64(4), int64(1)
memory usage: 53.7 KB
```

In [151]:
```
data.describe()
```

Out[151]:

	Variance	Skewness	Kurtosis	Entropy	Class
count	1372.000000	1372.000000	1372.000000	1372.000000	1372.000000
mean	0.433735	1.922353	1.397627	-1.191657	0.444606
std	2.842763	5.869047	4.310030	2.101013	0.497103
min	-7.042100	-13.773100	-5.286100	-8.548200	0.000000
25%	-1.773000	-1.708200	-1.574975	-2.413450	0.000000
50%	0.496180	2.319650	0.616630	-0.586650	0.000000
75%	2.821475	6.814625	3.179250	0.394810	1.000000
max	6.824800	12.951600	17.927400	2.449500	1.000000

In [152]:
```
import seaborn as sns
sns.countplot(x=data.Class, data=data['Class'].value_counts(), palette='hls')
plt.show()
```



In [153]:
```
from sklearn.model_selection import train_test_split
data_train, data_test = train_test_split(data, test_size=0.2, random_state=19)
```

In [154]:
```
print(data_train.shape)
print(data_test.shape)
```

```
(1097, 5)
(275, 5)
```

In [155]:
```
data_train.head(10)
```

Out[155]:

	Variance	Skewness	Kurtosis	Entropy	Class
528	3.624400	1.46090	1.35010	1.928400	0
896	0.004054	0.62905	-0.64121	0.758170	1
805	-3.608500	3.32530	-0.51954	-3.573700	1
762	-1.397100	3.31910	-1.39270	-1.994800	1
1065	-3.601200	-6.53890	10.52340	-0.489670	1
721	-0.450620	-1.36780	7.08580	-0.403030	0
532	2.522700	2.23690	2.72360	0.794380	0
1281	-2.790800	-5.71330	5.95300	0.459460	1
1173	-4.746200	3.12050	1.07500	-1.296600	1
558	4.384600	-4.87940	3.36620	-0.029324	0

In [156]:
```
X_train, y_train = data_train.drop('Class', axis=1), data_train['Class']
```

In [157]:
```
X_train.head()
```

Out[157]:

	Variance	Skewness	Kurtosis	Entropy
528	3.624400	1.46090	1.35010	1.92840
896	0.004054	0.62905	-0.64121	0.75817
805	-3.608500	3.32530	-0.51954	-3.57370
762	-1.397100	3.31910	-1.39270	-1.99480
1065	-3.601200	-6.53890	10.52340	-0.48967

In [158]:
```
y_train.head()
```

Out[158]:
```
528     0
896     1
805     1
762     1
1065    1
Name: Class, dtype: int64
```

In [159]:
```
X_test, y_test = data_test.drop('Class', axis=1), data_test['Class']
```

In [160]:
```
from sklearn.linear_model import LogisticRegression
```

In [161]:
```
clf = LogisticRegression().fit(X_train, y_train)
```

```
C:\Users\SR1407SM1106\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

In [162]:
```
clf.predict(X_test)
```

Out[162]:
```
array([1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1], dtype=int64)
```

In [163]:
```
data_pred = clf.predict(X_test)
clf.score(X_test, y_test)
```

Out[163]:
```
0.9927272727272727
```

In [164]:
```
df = pd.DataFrame({'Actual Class':y_test, 'Predicted Class':data_pred})
df
```

Out[164]:

	Actual Class	Predicted Class
1183	1	1
1010	1	1
1038	1	1
1341	1	1
225	0	0
...	...	...
19	0	0
481	0	0
110	0	0
1032	1	1
1314	1	1

275 rows × 2 columns

In [166]:
```
data_compare = np.where(df['Actual Class']== df['Predicted Class'], True, Fals
e)
df['Equal'] = data_compare
#df.drop('equal', axis=1, inplace=True)
df
```

Out[166]:

	Actual Class	Predicted Class	Equal
1183	1	1	True
1010	1	1	True
1038	1	1	True
1341	1	1	True
225	0	0	True
...	...	...	...
19	0	0	True
481	0	0	True
110	0	0	True
1032	1	1	True
1314	1	1	True

275 rows × 3 columns

In [167]:
```
df['Equal'].value_counts()
```

Out[167]:
```
True     273
False      2
Name: Equal, dtype: int64
```

Out of 273, 2 were predicted wrongly. Hence score is 99.2%

In [168]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, data_pred)
```

Out[168]:
```
array([[136,   2],
       [  0, 137]], dtype=int64)
```

In [169]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, data_pred))
```

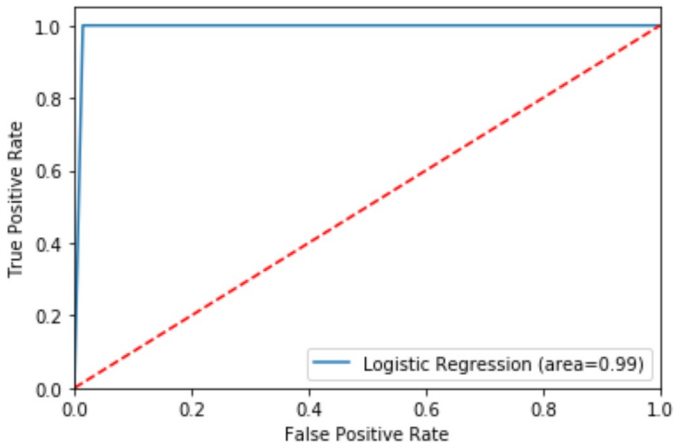
```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       138
           1       0.99      1.00      0.99       137

    accuracy                           0.99       275
   macro avg       0.99      0.99      0.99       275
weighted avg       0.99      0.99      0.99       275
```

In [170]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
```

In [171]:
```
roc_auc = roc_auc_score(y_test, data_pred)
fpr, tpr, thresholds = roc_curve(y_test, data_pred)
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area=%0.2f)' %
        roc_auc)
plt.plot([0,1], [0,1], 'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.show()
```



SVM

In [172]:
```
roc_auc_score(y_test, data_pred)
```

Out[172]:
```
0.9927536231884059
```

In [173]:
```
from sklearn import svm
```

In [174]:
```
clf = svm.SVC(kernel='linear')
clf.fit(X_train, y_train)
svm_pred = clf.predict(X_test)
clf.score(X_test, y_test)
```

Out[174]:
```
0.9927272727272727
```

In [175]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, svm_pred)
```

Out[175]:
```
array([[136,   2],
       [  0, 137]], dtype=int64)
```

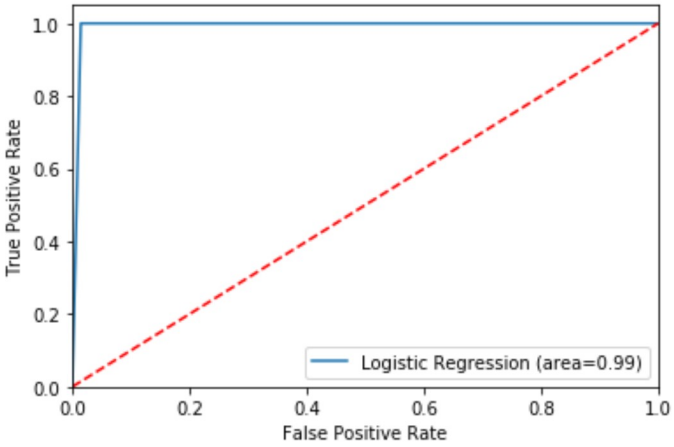
In [176]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, svm_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       138
           1       0.99      1.00      0.99       137

    accuracy                           0.99       275
   macro avg       0.99      0.99      0.99       275
weighted avg       0.99      0.99      0.99       275
```

In [177]:
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
roc_auc = roc_auc_score(y_test, svm_pred)
fpr, tpr, thresholds = roc_curve(y_test, svm_pred)
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area=%0.2f)' %
        roc_auc)
plt.plot([0,1], [0,1], 'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc='lower right')
plt.show()
```



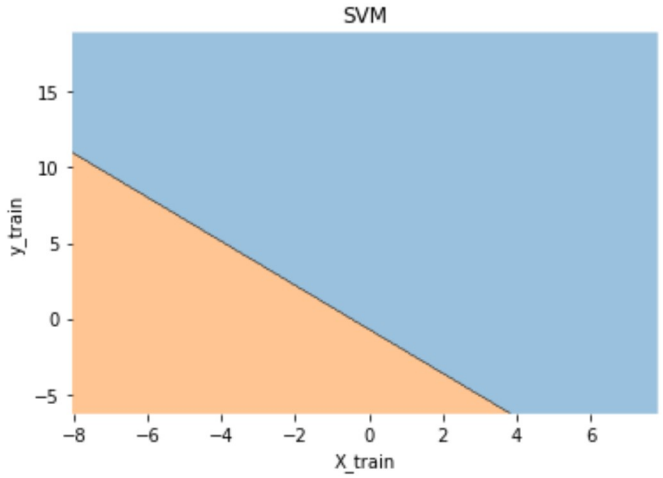
In [178]:
```
X_train[['Variance', 'Skewness']]
```

Out[178]:

	Variance	Skewness
528	3.624400	1.46090
896	0.004054	0.62905
805	-3.608500	3.32530
762	-1.397100	3.31910
1065	-3.601200	-6.53890
...	...	...
308	4.616000	10.17880
1043	-2.702800	1.63270
936	-1.278600	-2.40870
757	2.660600	3.16810
622	5.042900	-0.52974

1097 rows × 2 columns

In [179]:
```
from mlxtend.plotting import plot_decision_regions
plot_decision_regions(X_train.to_numpy(), y_train.to_numpy(), clf=clf, legend=
2, feature_index=[0,2], filler_feature_values={1:2, 3:4})
plt.xlabel('X_train')
plt.ylabel('y_train')
plt.title('SVM')
plt.show()
```



unable to plot decision regions

In [ ]: