Yuli Kim
CS 634
Final Project
Github link: https://github.com/yk348/final_term_proj

Data was obtained from the UC Irvine Machine Learning Repository. The objective of this project is to compare three different algorithms. For the project, I used Random forest, SVM, and LSTM.

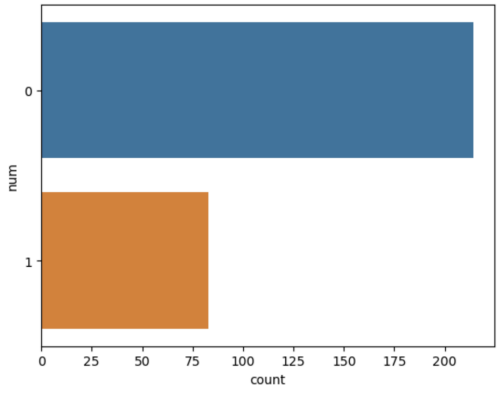
Random Forest

]:

	TP	TN	FP	FN	TPR	TNR	FPR	FNR	Precision	F1_measure	Accuracy	Error_rate	BACC	TSS	HSS
0	4	21	3	2	0.666667	0.875000	0.125000	0.333333	0.571429	0.615385	0.833333	0.166667	0.770833	0.541667	0.509804
1	5	20	2	3	0.625000	0.909091	0.090909	0.375000	0.714286	0.666667	0.833333	0.166667	0.767045	0.534091	0.556213
2	3	20	4	3	0.500000	0.833333	0.166667	0.500000	0.428571	0.461538	0.766667	0.233333	0.666667	0.333333	0.313725
3	7	16	5	2	0.777778	0.761905	0.238095	0.222222	0.583333	0.666667	0.766667	0.233333	0.769841	0.539683	0.492754
4	6	21	1	2	0.750000	0.954545	0.045455	0.250000	0.857143	0.800000	0.900000	0.100000	0.852273	0.704545	0.733728
5	6	17	2	5	0.545455	0.894737	0.105263	0.454545	0.750000	0.631579	0.766667	0.233333	0.720096	0.440191	0.467005
6	4	21	5	0	1.000000	0.807692	0.192308	0.000000	0.444444	0.615385	0.833333	0.166667	0.903846	0.807692	0.528302
7	3	21	4	1	0.750000	0.840000	0.160000	0.250000	0.428571	0.545455	0.827586	0.172414	0.795000	0.590000	0.448669
8	3	18	5	3	0.500000	0.782609	0.217391	0.500000	0.375000	0.428571	0.724138	0.275862	0.641304	0.282609	0.251613
9	9	18	2	0	1.000000	0.900000	0.100000	0.000000	0.818182	0.900000	0.931034	0.068966	0.950000	0.900000	0.848168

]:
```
for train_index, test_index in kf.split(X):
  #  print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    rf.fit(X_train,y_train)
    y_pred = rf.predict(X_test)

    TN, FP, FN, TP = confusion_matrix(y_pred,np.array(y_test)).ravel()
    TPR = TP / (TP + FN)
    TNR = TN / (TN + FP)
    FPR = FP / (TN + FP)
    FNR = FN / (TP + FN)
    Precision = TP / (TP + FP)
    F1_measure = 2 * TP / (2 * TP + FP + FN)
    Accuracy = (TP + TN) / (TP + FP + FN + TN)
    Error_rate = (FP + FN) / (TP + FP + FN + TN)
    BACC = (TPR + TNR) / 2
    TSS = TPR - FPR
    HSS = 2 * (TP * TN - FP * FN) / ((TP + FN) * (FN + TN) + (TP + FP) * (FP + TN))

    metrics = [TP, TN, FP, FN, TPR, TNR, FPR, FNR, Precision, F1_measure, Accuracy, Error_rate, BACC, TSS, HSS]

    test_results.append([TP, TN, FP, FN, TPR, TNR, FPR, FNR, Precision, F1_measure, Accuracy, Error_rate, BACC, TSS, HSS])
```

]:
```
test_result = pd.DataFrame(test_results, columns=['TP', 'TN', 'FP', 'FN', 'TPR', 'TNR', 'FPR', 'FNR', 'Precision', 'F1_measure', 'Accuracy', '
test_result
```



SVM

	TP	TN	FP	FN	TPR	TNR	FPR	FNR	Precision	F1_measure	Accuracy	Error_rate	BACC	TSS	HSS
0	0	23	7	0	NaN	0.766667	0.233333	NaN	0.0	0.0	0.766667	0.233333	NaN	NaN	0.0
1	0	23	7	0	NaN	0.766667	0.233333	NaN	0.0	0.0	0.766667	0.233333	NaN	NaN	0.0
2	0	23	7	0	NaN	0.766667	0.233333	NaN	0.0	0.0	0.766667	0.233333	NaN	NaN	0.0
3	0	18	12	0	NaN	0.600000	0.400000	NaN	0.0	0.0	0.600000	0.400000	NaN	NaN	0.0
4	0	23	7	0	NaN	0.766667	0.233333	NaN	0.0	0.0	0.766667	0.233333	NaN	NaN	0.0
5	0	22	8	0	NaN	0.733333	0.266667	NaN	0.0	0.0	0.733333	0.266667	NaN	NaN	0.0
6	0	21	9	0	NaN	0.700000	0.300000	NaN	0.0	0.0	0.700000	0.300000	NaN	NaN	0.0
7	0	22	7	0	NaN	0.758621	0.241379	NaN	0.0	0.0	0.758621	0.241379	NaN	NaN	0.0
8	0	21	8	0	NaN	0.724138	0.275862	NaN	0.0	0.0	0.724138	0.275862	NaN	NaN	0.0
9	0	18	11	0	NaN	0.620690	0.379310	NaN	0.0	0.0	0.620690	0.379310	NaN	NaN	0.0

```
from sklearn.svm import SVC
```

```
test_results = []
svm = SVC()
for train_index, test_index in kf.split(X):
  #  print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    svm.fit(X_train,y_train)
    y_pred = svm.predict(X_test)

    TN, FP, FN, TP = confusion_matrix(y_pred,np.array(y_test)).ravel()
    TPR = TP / (TP + FN)
    TNR = TN / (TN + FP)
    FPR = FP / (TN + FP)
    FNR = FN / (TP + FN)
    Precision = TP / (TP + FP)
    F1_measure = 2 * TP / (2 * TP + FP + FN)
    Accuracy = (TP + TN) / (TP + FP + FN + TN)
    Error_rate = (FP + FN) / (TP + FP + FN + TN)
    BACC = (TPR + TNR) / 2
    TSS = TPR - FPR
    HSS = 2 * (TP * TN - FP * FN) / ((TP + FN) * (FN + TN) + (TP + FP) * (FP + TN))

    metrics = [TP, TN, FP, FN, TPR, TNR, FPR, FNR, Precision, F1_measure, Accuracy, Error_rate, BACC, TSS, HSS]

    test_results.append([TP, TN, FP, FN, TPR, TNR, FPR, FNR, Precision, F1_measure, Accuracy, Error_rate, BACC, TSS, HSS])
```

```
test_result = pd.DataFrame(test_results, columns=['TP', 'TN', 'FP', 'FN', 'TPR', 'TNR', 'FPR', 'FNR', 'Precision', 'F1_measure', 'Accuracy', '
test_result
```

LSTM

]:

	TP	TN	FP	FN	TPR	TNR	FPR	FNR	Precision	F1_measure	Accuracy	Error_rate	BACC	TSS	HSS
0	0	23	7	0	NaN	0.766667	0.233333	NaN	0.000000	0.000000	0.766667	0.233333	NaN	NaN	0.000000
1	0	23	7	0	NaN	0.766667	0.233333	NaN	0.000000	0.000000	0.766667	0.233333	NaN	NaN	0.000000
2	0	22	7	1	0.000000	0.758621	0.241379	1.000000	0.000000	0.000000	0.733333	0.266667	0.379310	-0.241379	-0.061947
3	11	16	1	2	0.846154	0.941176	0.058824	0.153846	0.916667	0.880000	0.900000	0.100000	0.893665	0.787330	0.794521
4	7	20	0	3	0.700000	1.000000	0.000000	0.300000	1.000000	0.823529	0.900000	0.100000	0.850000	0.700000	0.756757
5	5	15	3	7	0.416667	0.833333	0.166667	0.583333	0.625000	0.500000	0.666667	0.333333	0.625000	0.250000	0.264706
6	0	21	9	0	NaN	0.700000	0.300000	NaN	0.000000	0.000000	0.700000	0.300000	NaN	NaN	0.000000
7	4	20	3	2	0.666667	0.869565	0.130435	0.333333	0.571429	0.615385	0.827586	0.172414	0.768116	0.536232	0.505119
8	0	21	8	0	NaN	0.724138	0.275862	NaN	0.000000	0.000000	0.724138	0.275862	NaN	NaN	0.000000
9	1	18	10	0	1.000000	0.642857	0.357143	0.000000	0.090909	0.166667	0.655172	0.344828	0.821429	0.642857	0.110429

```
test_results = []
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(n_steps, n_features)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

  #  print("TRAIN:", train_index, "TEST:", test_index)
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    TN, FP, FN, TP = confusion_matrix(y_pred,np.array(y_test)).ravel()
    TPR = TP / (TP + FN)
    TNR = TN / (TN + FP)
    FPR = FP / (TN + FP)
    FNR = FN / (TP + FN)
    Precision = TP / (TP + FP)
    F1_measure = 2 * TP / (2 * TP + FP + FN)
    Accuracy = (TP + TN) / (TP + FP + FN + TN)
    Error_rate = (FP + FN) / (TP + FP + FN + TN)
    BACC = (TPR + TNR) / 2
    TSS = TPR - FPR
    HSS = 2 * (TP * TN - FP * FN) / ((TP + FN) * (FN + TN) + (TP + FP) * (FP + TN))

    metrics = [TP, TN, FP, FN, TPR, TNR, FPR, FNR, Precision, F1_measure, Accuracy, Error_rate, BACC, TSS, HSS]

    test_results.append([TP, TN, FP, FN, TPR, TNR, FPR, FNR, Precision, F1_measure, Accuracy, Error_rate, BACC, TSS, HSS])
```

Based on the graphs, we can see that the random forest algorithm method has the highest accuracy and precision. While SVM and LSTM have similar results, SVM out performed LSTM.