Poker Rule Induction

By: Daniel Arroyo, Emanoel Khachadorian, Mher Oganesyan

We The Algorithm,

- Learn the rules of poker.
- Develop an algorithm that can learn to play poker by analyzing outcome of previously played hands.
- Then predicting the best hand we can play based on the cards we've been dealt.

Data Details

- The training data has 10 features and 1 label.
- Suits of cards are labeled 1-4 each representing, Hearts, Spades, Diamonds, Clubs respectively.
- The rank of cards are numbered from 1-13 and they represent Ace, 2 through 10, Jack, Queen and King.

Snapshot of Data

	A	B	C	D	E	F	G	H	I	J	K
1	S1	C1	S2	C2	S3	C3	S4	C4	S5	C5	hand
2	4	9	2	1	2	2	4	7	2	8	0
3	1	4	3	6	1	12	3	11	2	7	0
4	1	11	4	1	3	7	4	11	2	1	2
5	2	9	2	4	3	6	1	9	4	9	3
6	1	8	2	4	2	11	2	2	2	1	0
7	2	5	1	5	2	13	2	3	3	13	2
8	3	10	4	6	1	4	2	13	4	5	0
9	4	10	3	1	2	13	4	2	4	7	0
10	3	2	4	10	3	3	4	4	1	9	0
11	2	7	3	8	4	8	2	13	2	12	1
12	2	5	1	3	2	10	3	2	2	1	0
13	1	6	2	12	4	7	2	10	1	1	0
14	4	2	4	9	1	12	3	7	2	11	0
15	2	6	1	5	3	3	4	2	4	5	1
16	1	6	3	12	4	11	2	11	3	13	1
17	2	5	2	4	4	9	2	3	3	2	0
18	4	9	4	11	3	8	3	9	3	5	1
19	1	9	2	4	1	11	3	4	1	13	1
20	1	11	3	13	4	8	4	1	3	6	0
21	2	6	1	12	3	8	4	1	4	6	1
22	2	9	3	7	1	13	2	13	1	2	1
23	4	4	2	13	3	6	2	7	3	3	0
24	3	6	1	4	3	1	2	5	4	9	0
25	1	4	4	3	2	11	4	13	1	10	0
26	3	9	2	7	2	4	2	1	1	10	0
27	2	4	4	2	1	4	4	1	3	11	1
28	2	8	1	4	3	11	1	8	1	2	1
29	3	8	1	12	4	1	4	10	1	3	0
30	2	12	1	5	1	2	1	10	2	8	0

Developed Methods and Algorithms

- Algorithms Implemented
  - K-Nearest Neighbors
  - Decision Tree
  - Logistic Regression
  - Random Forest
- Methods Applied
  - Cross Validation
  - One Hot Encoding
  - Custom feature engineering

Results prior to feature engineering

K-Nearest Neighbors

- Accuracy without Cross Validation
  - 53.8 Percent
- Accuracy with Cross Validation
  - 52.9 Percent

KNN

```
k = 3
my_knn = KNeighborsClassifier(n_neighbors=k)
my_knn.fit(X_train, y_train)
y_predict = my_knn.predict(X_test)
accuracy = accuracy_score(y_test, y_predict)

print(accuracy)
```

```
0.5375019990404606
```

```
accuracy_list = cross_val_score(my_knn, X, y, cv=5, scoring='accuracy')
accuracy_list_dict["accuracy_list_KNN"] = accuracy_list
accuracy_list_dict["accuracy_cv_KNN"] = accuracy_list.mean()
```

Results prior to feature engineering

Decision Tree

- Accuracy without Cross Validation
  - 49 Percent
- Accuracy with Cross Validation
  - 49 Percent

Decision Tree

```
my_decisiontree = DecisionTreeClassifier()
my_decisiontree.fit(X_train, y_train)
y_predict = my_decisiontree.predict(X_test)
accuracy = accuracy_score(y_test, y_predict)
```

```
print(accuracy)
```

```
0.4896849512234127
```

```
accuracy_list = cross_val_score(my_decisiontree, X, y, cv=5, scoring='accuracy')
accuracy_list_dict["accuracy_list_DT"] = accuracy_list
accuracy_list_dict["accuracy_cv_DT"] = accuracy_list.mean()
```

Results prior to feature engineering

Logistic Regression

- Accuracy without Cross Validation
    - 50.5 Percent
- Accuracy with Cross Validation
    - 50 Percent

Logistic Regression

```
my_logreg = LogisticRegression()
my_logreg.fit(X_train, y_train)
y_predict_lr = my_logreg.predict(X_test)
accuracy = accuracy_score(y_test, y_predict_lr)

print(accuracy)
```

```
0.5053574284343515
```

```
accuracy_list = cross_val_score(my_logreg, X, y, cv=5, scoring='accuracy')
accuracy_list_dict["accuracy_list_LR"] = accuracy_list
accuracy_list_dict["accuracy_cv_LR"] = accuracy_list.mean()
```

Results prior to feature engineering

Random Forest

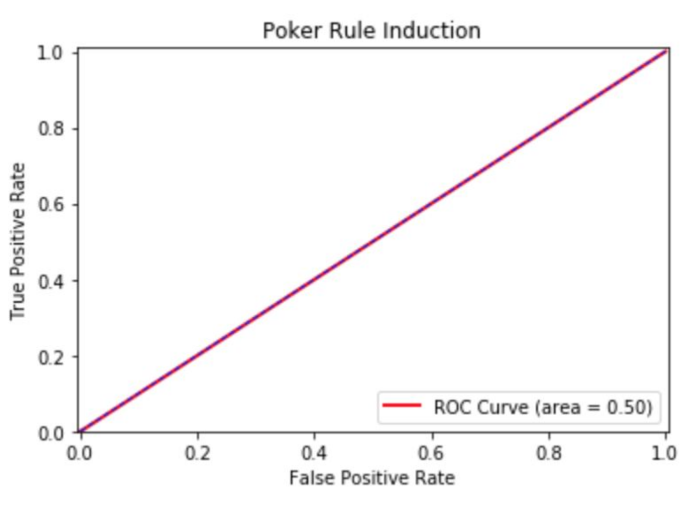
- Accuracy without Cross Validation
  - 57 Percent
- Accuracy with Cross Validation
  - 57.7 Percent

```
from sklearn.ensemble import RandomForestClassifier
my_RandomForest = RandomForestClassifier(n_estimators = 19, bootstrap = True, random_state=3)
my_RandomForest.fit(X_train, y_train)
y_predict_rf = my_RandomForest.predict(X_test)
accuracy = accuracy_score(y_test, y_predict_rf)
print(accuracy)
```

```
accuracy_list = cross_val_score(my_RandomForest, X, y, cv=5, scoring='accuracy')
accuracy_list_dict["accuracy_list_RF"] = accuracy_list
accuracy_list_dict["accuracy_cv_RF"] = accuracy_list.mean()
```

Results with Naive feature engineering

- Measuring Area Under Curve with One Hot Encoding
  - Applied to Random Forest because of its higher initial accuracy
    - Cross Validation: 61.4 Percent
    - Area Under Curve: 50 percent



Feature engineering on suits

- Count how many of each suit in the hand
- This made information for each suit more meaningful
- Allowed for easier recognizing of hand types that relied on suits, like flush, straight flush, royal flush
- Example: if there is a 5 for any suit type then only one of the above three flush types are possible

hearts	spades	diamonds	clubs
2	1	2	0
1	1	2	1
2	3	0	0
2	0	2	1
1	3	1	0
2	2	1	0
2	0	3	0
2	2	1	0
2	0	3	0
2	1	2	0

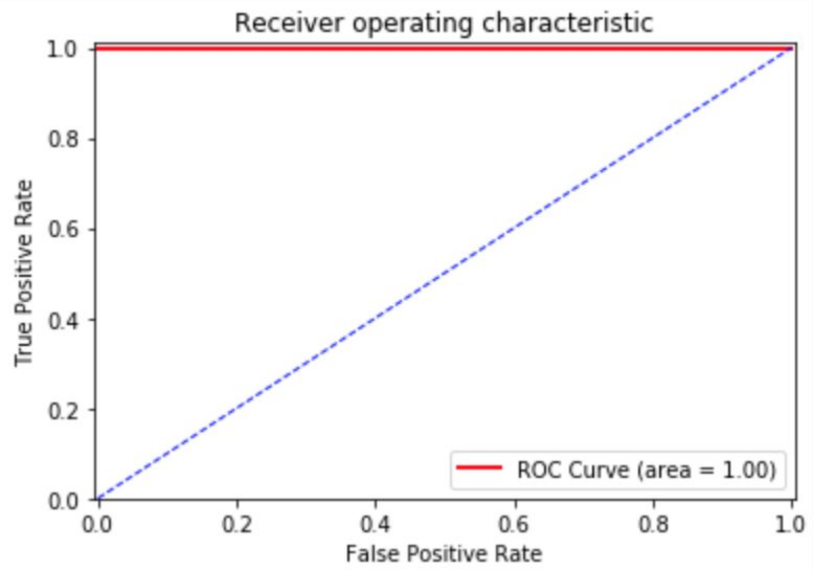
Feature engineering on ranks

- Instead of having the rank of each card, the cards were sorted and the difference between the neighboring cards were taken
- This made it easier to recognize when cards were the same, difference of 0, or when they were in order, difference of 1.
- Example: any combination of 0-n-0 is two pairs and 1-1-1-1-1 is straight/straight flush

one2two	two2three	three2four	four2five	five2one
1	1	5	2	9
1	1	1	6	9
2	1	4	3	10
3	1	1	3	8
2	0	4	2	8
2	1	3	1	7
4	1	1	1	7
2	5	3	0	10
2	1	1	1	5
1	1	7	2	11
6	1	2	1	10
2	4	1	1	8

Results on training file only

- 75 percent training set
- 25 percent testing set
- Decision Tree
  - Accuracy: 0.99968
  - AUC: 1.0
- Random Forest
  - Accuracy: 0.99968
  - AUC: 1.0



Kaggle Submission Result

- 25,000 training hands
- 1,000,000 testing hands
- Decision tree
  - Accuracy of .99995 (50 wrong hands)
- Random Forest
  - Accuracy of .99889 (1,110 wrong hands)

Name	Submitted	Wait time	Execution time	Score
TestResults.csv	a few seconds ago	0 seconds	7 seconds	0.99995

Complete

Jump to your position on the leaderboard