1. implement linear SVM algorithm on final boards classification and single label classification.
2. repeat for KNN and multilayer perceptron.
3. write a single program that outputs accuracy and confusion matrices for both datasets and for all the classifiers.

[8]:
```
# import libraries
import pandas as pd
import numpy as np
import sklearn
# dataset paths
final_txt_path = "./datasets/datasets-part1/tictac_final.txt"
multi_txt_path = "./datasets/datasets-part1/tictac_multi.txt"
single_txt_path = "./datasets/datasets-part1/tictac_single.txt"
```

[9]:
```
from sklearn.model_selection import train_test_split

def get_test_train(path):
    data_set = open(path,"r")
    # build x and y
    X = []
    y = []
    for line in data_set:
        temp = line.rstrip().split(" ")
        X.append(temp[:9])
        if len(temp) == 10:
            y.append(temp[-1])
        else:
            y.append(temp[-9:])
    # split into train and test
    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,shuffle=True)
    return X_train, X_test, y_train, y_test
```

Classifier Functions

[10]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# create function to get accuracy and plot confusion matrix
def get_stats(actual, pred, title, cmap=plt.cm.gray_r):
    accuracy = sklearn.metrics.accuracy_score(actual, pred, normalize=True)

    a = pd.Series(actual, name='Actual')
    p = pd.Series(pred, name='Predicted')
    df_confusion = pd.crosstab(a, p)

    plt.matshow(df_confusion, cmap=cmap) # imshow
    plt.colorbar()
    tick_marks = np.arange(len(df_confusion.columns))
    plt.xticks(tick_marks, df_confusion.columns, rotation=45)
    plt.yticks(tick_marks, df_confusion.index)
    plt.ylabel(df_confusion.index.name)
    plt.xlabel(df_confusion.columns.name)
    title = title + f" (Accuracy: {round(accuracy, 4)})"
    plt.title(title)
    plt.show()
```

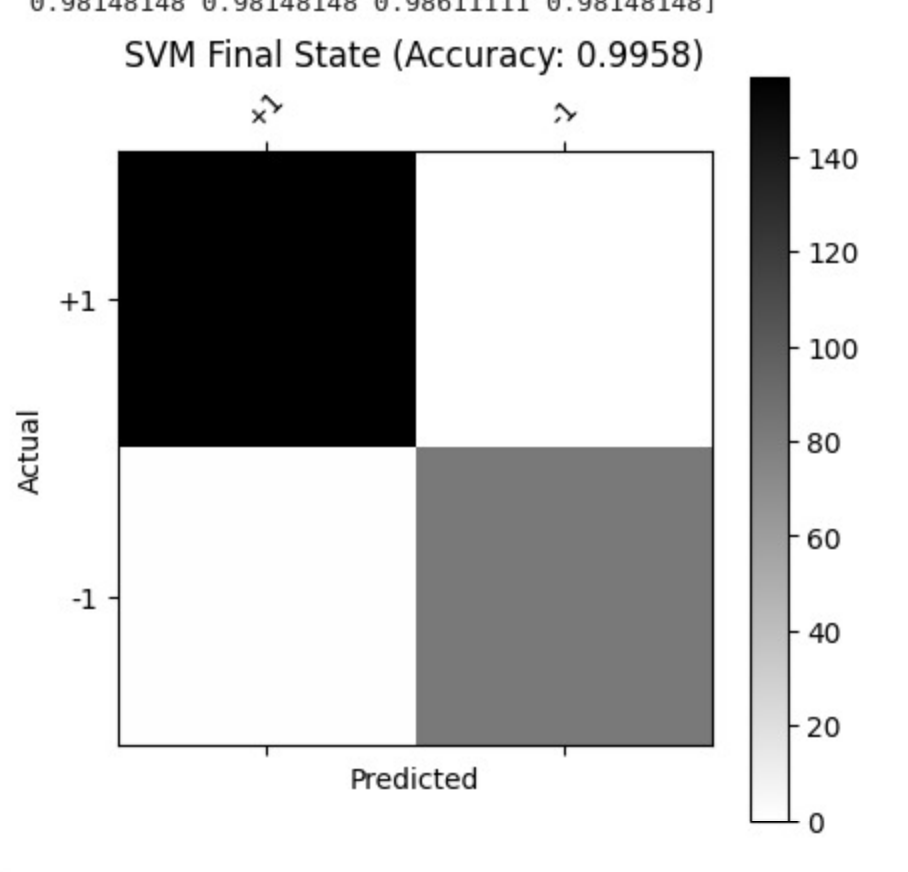
[11]:
```
from sklearn.model_selection import ShuffleSplit, cross_val_score

def clf_pipeline(model, path, title, transform_X=False):
    # get testing data
    X_train, X_test, y_train, y_test = get_test_train(path)
    if transform_X:
        X_train = np.array(X_train, dtype=np.float64)
        X_test = np.array(X_test, dtype=np.float64)
    # get cross-validation score
    cv = ShuffleSplit(n_splits=10, test_size=0.3, random_state=0)
    score = cross_val_score(model, X_train, y_train, cv=cv)
    print(f"Cross Validation Scores:\n{score}")
    # fit model
    model.fit(X_train, y_train)
    y_hat = model.predict(X_test)
    # print out stats
    get_stats(y_test, y_hat, title)
```

Classification: Linear SVM - Final Board State and Optimal Single Move

[12]:
```
from sklearn import svm, metrics
clf = svm.SVC()
```

[13]:
```
clf_pipeline(clf, final_txt_path, "SVM Final State")
```

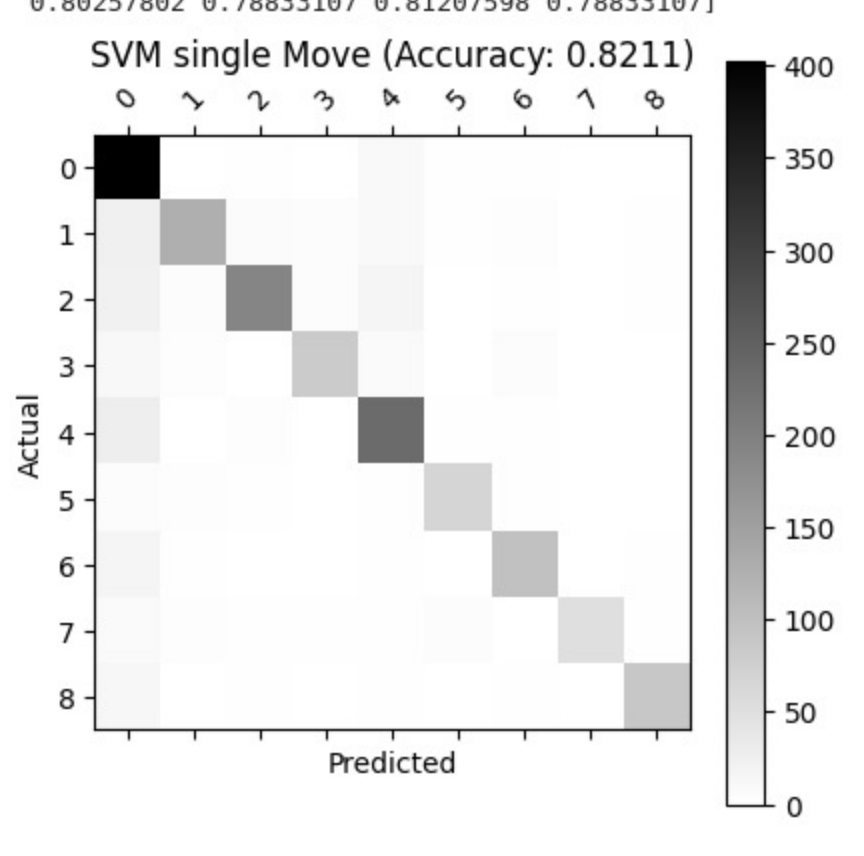
```
Cross Validation Scores:
[0.98148148 0.96759259 0.98611111 0.99074074 0.98611111 0.98611111
 0.98148148 0.98148148 0.98611111 0.98148148]
```



[14]:
```
# build model
clf = svm.SVC()
```

[15]:
```
clf_pipeline(clf, single_txt_path, "SVM single Move")
```

```
Cross Validation Scores:
[0.78426052 0.78561737 0.78086839 0.79579376 0.78629579 0.7917232
 0.80257802 0.78833107 0.81207598 0.78833107]
```

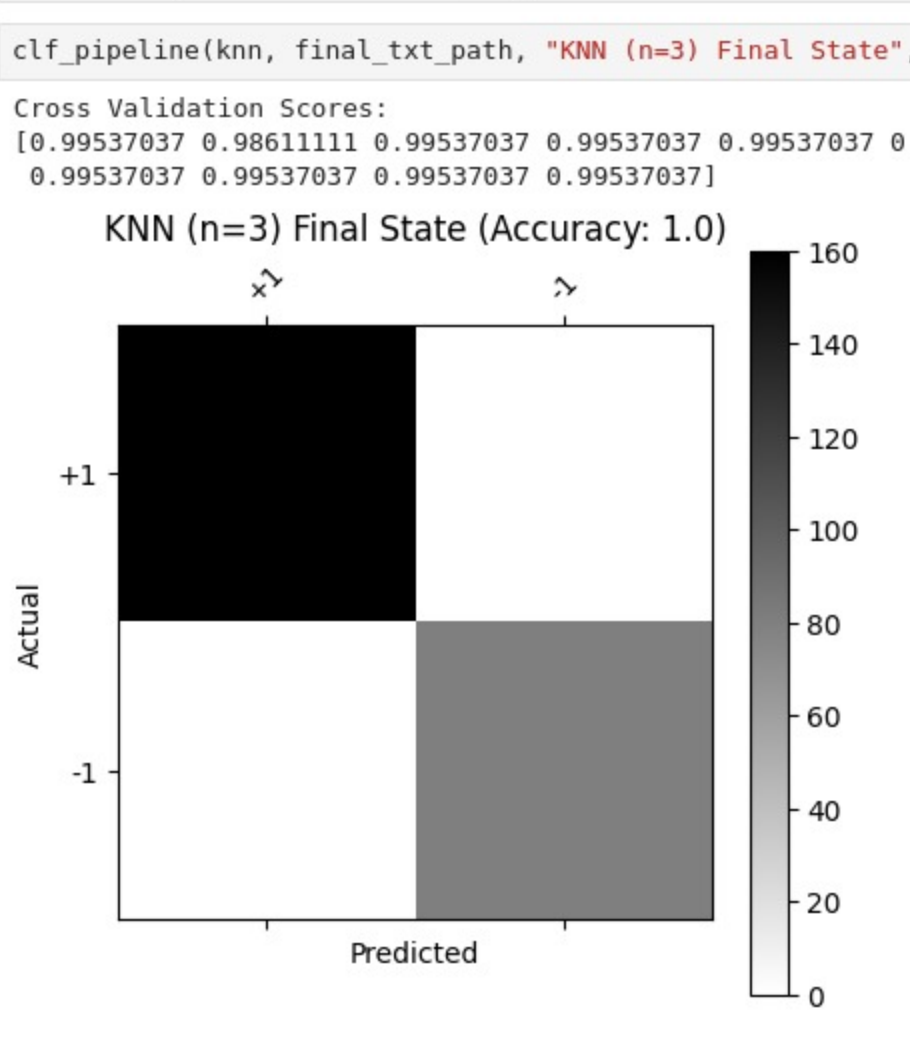


Classification: KNN - Final Board State and Optimal Single Move

[16]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
```

[17]:
```
clf_pipeline(knn, final_txt_path, "KNN (n=3) Final State", transform_X=True)
```

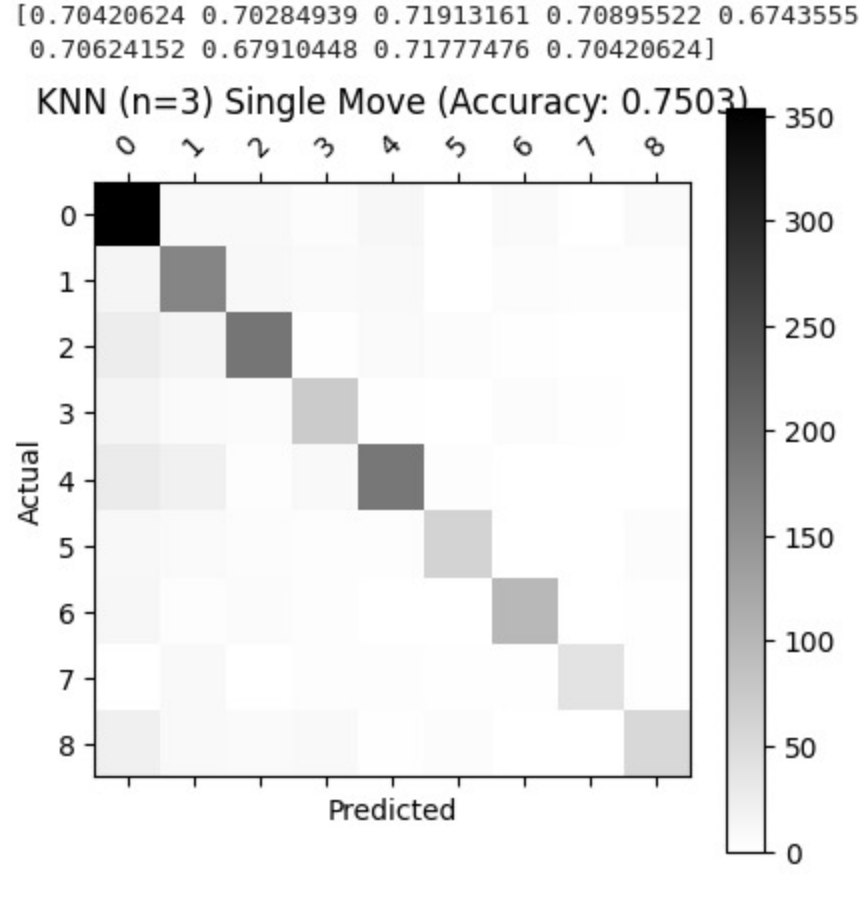
```
Cross Validation Scores:
[0.99537037 0.98611111 0.99537037 0.99537037 0.99537037 0.99537037
 0.99537037 0.99537037 0.99537037 0.99537037]
```



[18]:
```
knn = KNeighborsClassifier(n_neighbors=3)
```

[19]:
```
clf_pipeline(knn, single_txt_path, "KNN (n=3) Single Move", transform_X=True)
```

```
Cross Validation Scores:
[0.70420624 0.70284939 0.71913161 0.70895522 0.6743555  0.69810041
 0.70624152 0.67910448 0.71777476 0.70420624]
```

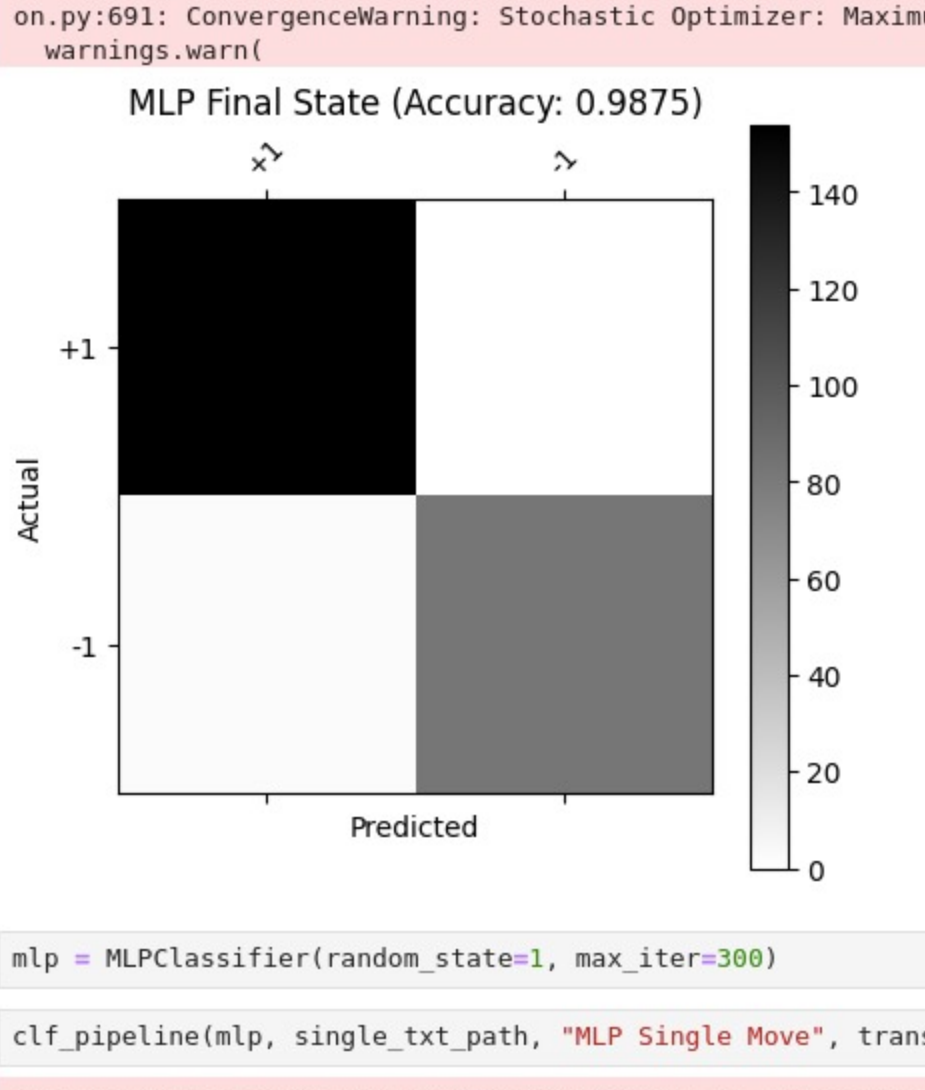


Classification: MLP - Final Board State and Optimal Single Move

[20]:
```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(random_state=1, max_iter=300)
```

[21]:
```
clf_pipeline(mlp, final_txt_path, "MLP Final State", transform_X=True)
```

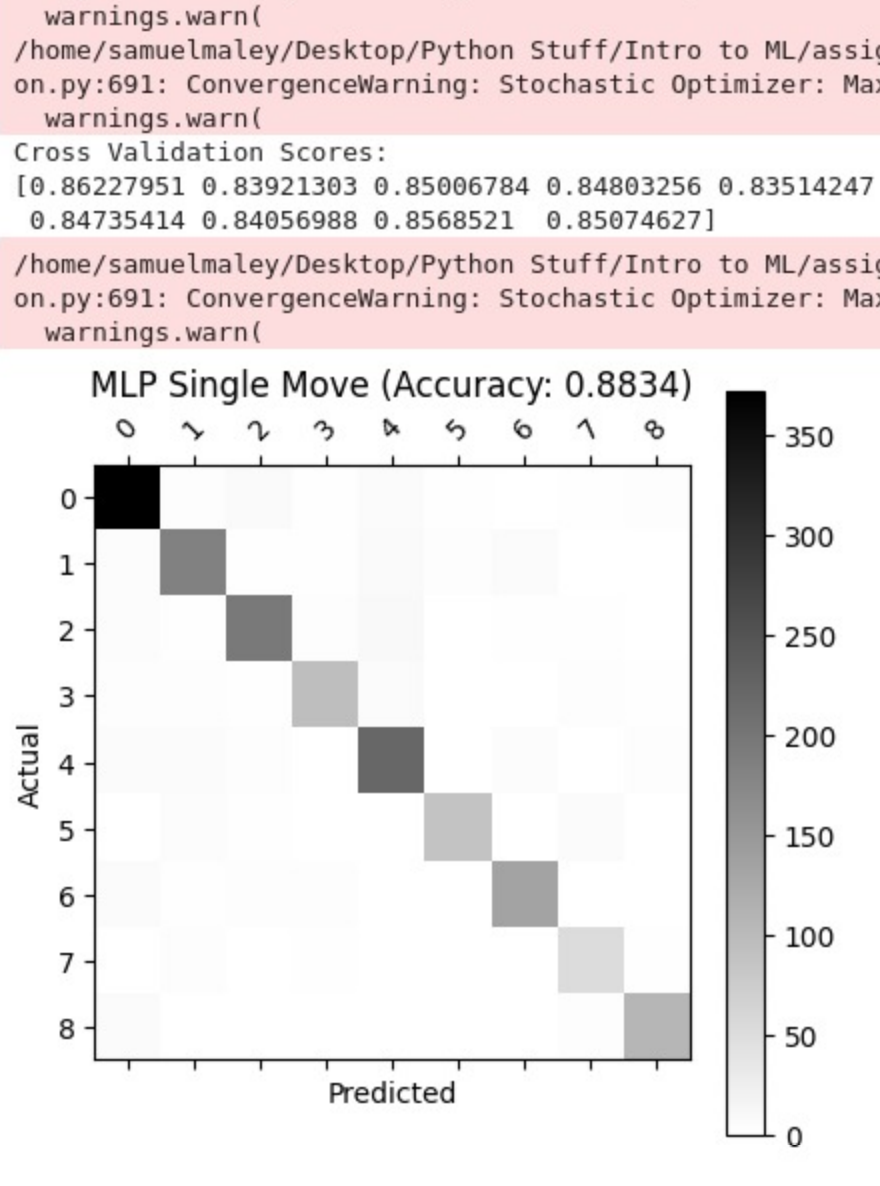
```
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
Cross Validation Scores:
[0.99074074 0.97685185 0.98611111 0.98148148 0.99074074 0.99074074
 0.97685185 0.97685185 0.99074074 0.97685185]
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
```



[22]:
```
mlp = MLPClassifier(random_state=1, max_iter=300)
```

[23]:
```
clf_pipeline(mlp, single_txt_path, "MLP Single Move", transform_X=True)
```

```
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
Cross Validation Scores:
[0.86227951 0.83921303 0.85006784 0.84803256 0.83514247 0.85820896
 0.84735414 0.84056988 0.8568521  0.85074627]
/home/samuelmaley/Desktop/Python Stuff/Intro to ML/assignment1/.venv/lib/python3.10/site-packages/sklearn/neural_network/_multilayer_perceptr
on.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (300) reached and the optimization hasn't converged yet.
  warnings.warn(
```



Regression Functions

[24]:
```
from sklearn.metrics import root_mean_squared_error

def get_reg_stats(actual, pred):
    accuracy = sklearn.metrics.accuracy_score(actual, pred, normalize=True)
    print(f"Accuracy:\n{accuracy}")
    rmse = root_mean_squared_error(actual, pred)
    print(f"RMSE:\n{round(rmse,4)}")
```

[39]:
```
from sklearn.model_selection import ShuffleSplit, cross_val_score

def reg_pipeline(model, path, transform=True):
    # get testing data
    X_train, X_test, y_train, y_test = get_test_train(path)
    if transform:
        X_train = np.array(X_train, dtype=np.float64)
        X_test = np.array(X_test, dtype=np.float64)
        y_train = np.array(y_train, dtype=np.float64)
        y_test = np.array(y_test, dtype=np.float64)
    # get cross-validation score
    cv = ShuffleSplit(n_splits=10, test_size=0.3, random_state=0)
    score = cross_val_score(model, X_train, y_train, cv=cv)
    print(f"Cross Validation Scores:\n{score}")
    # fit model
    model.fit(X_train, y_train)
    y_hat = model.predict(X_test)
    for arr in y_hat:
        for i in range(len(arr)):
            if arr[i] >= 0.5:
                arr[i] = 1
            else:
                arr[i] = 0
    print(y_hat)
    # print out stats
    get_reg_stats(y_test, y_hat)
```

Regression: KNN - Optimal Multi Move

[26]:
```
from sklearn.neighbors import KNeighborsRegressor
knn_reg = KNeighborsRegressor(n_neighbors=2,weights="distance")
```

[27]:
```
reg_pipeline(knn_reg, multi_txt_path)
```

```
Cross Validation Scores:
[0.55140073 0.5764239  0.58052073 0.57606772 0.58439506 0.58676308
 0.55549314 0.56016051 0.57509091 0.57254055]
Accuracy:
0.7118437118437119
RMSE:
0.2699
```

[28]:
```
from joblib import dump, load
dump(knn_reg, 'knn_reg.joblib')
```

[28]: ['knn_reg.joblib']

Regression: Linear - Optimal Multi Move

[44]:
```
from sklearn.multioutput import MultiOutputRegressor
from sklearn.linear_model import LinearRegression
l_reg = MultiOutputRegressor(LinearRegression())
```

[45]:
```
reg_pipeline(l_reg, multi_txt_path)
```

```
Cross Validation Scores:
[-7.21398483e-06 -3.35143319e-03  9.31764264e-04  2.48391542e-03
  1.23314670e-04  2.59757541e-03  1.99806882e-04 -3.15529650e-04
  7.52969683e-05  1.33386744e-03]
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
Accuracy:
0.0
RMSE:
0.4672
```

[33]:
```
from joblib import dump, load
dump(l_reg, 'l_reg.joblib')
```

[33]: ['l_reg.joblib']

Regression: MLP - Optimal Multi Move

[11]:
```
from sklearn.neural_network import MLPRegressor
mlp_reg = MLPRegressor(hidden_layer_sizes=(400,400), max_iter=1000)
```

[12]:
```
reg_pipeline(mlp_reg, multi_txt_path)
```

```
Cross Validation Scores:
[0.70662267 0.72158933 0.72863184 0.73376237 0.73379345 0.71454103
 0.71692388 0.71859324 0.72387904 0.71556909]
Accuracy:
0.8376068376068376
RMSE:
0.1654
```

[14]:
```
from joblib import dump, load
dump(mlp_reg, 'mlp_reg.joblib')
```

[14]: ['mlp_reg.joblib']

[29]:
```
mlp_reg2 = load('mlp_reg.joblib')
```

[30]:
```
X_train, X_test, y_train, y_test = get_test_train(multi_txt_path)
X_train = np.array(X_train, dtype=np.float64)
X_test = np.array(X_test, dtype=np.float64)
y_train = np.array(y_train, dtype=np.float64)
y_test = np.array(y_test, dtype=np.float64)
```

[26]:
```
print(X_test)
```

```
[[ 1. -1.  1. ...  0.  0. -1.]
 [-1.  1. -1. ...  1.  0.  0.]
 [ 1.  0.  0. ...  0.  0.  0.]
 ...
 [ 0.  1. -1. ...  1.  0.  0.]
 [-1.  0.  1. ... -1.  1.  0.]
 [-1.  1.  1. ...  0. -1. -1.]]
```

[23]:
```
y_pred = mlp_reg2.predict(X_test)
```

[24]:
```
for arr in y_pred:
    for i in range(len(arr)):
        if arr[i] >= 0.5:
            arr[i] = 1
        else:
            arr[i] = 0
```

[25]:
```
accuracy = sklearn.metrics.accuracy_score(y_test, y_pred, normalize=True)
print(f"Accuracy:\n{accuracy}")
rmse = root_mean_squared_error(y_test, y_pred)
print(f"RMSE:\n{round(rmse,4)}")
```

```
Accuracy:
0.9597069597069597
RMSE:
0.0841
```

[ ]: