In [1]:

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

from sklearn.model_selection import cross_val_score, GridSearchCV, train_test_split
from sklearn.feature_selection import SelectFromModel

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier

import matplotlib.pyplot as plt
```

In [2]:

```
#Import soccer data(database.sqlite) using pandas.
import pandas as pd
import sqlite3
db = 'database.sqlite'
conn = sqlite3.connect(db)
cur = conn.cursor()
```

In [3]:

```
#Load the values from the attributes X= ['strength', 'stamina'] and Y= ['jumping'] from

df = pd.read_sql("SELECT strength, stamina, jumping FROM Player_Attributes", conn)
df.fillna(11, inplace=True)

df.head()
```

Out[3]:

	strength	stamina	jumping
0	76.0	54.0	58.0
1	76.0	54.0	58.0
2	76.0	54.0	58.0
3	76.0	54.0	58.0
4	76.0	54.0	58.0

In [4]:

```
x = df[['strength', 'stamina']].values
y = df[['jumping']].values
X_train, X_test, y_train, y_test = train_test_split(x, y)
```

In [5]:

```
#Using the above X and Y, apply GridSearch on DecisionTreeClassifier and fetch the best
from sklearn import metrics

model = DecisionTreeClassifier()
dt_mod = model.fit(X_train, y_train)
dt_preds = dt_mod.predict(X_test)
accuracy = metrics.accuracy_score(y_test, dt_preds)
accuracy
```

Out[5]:

```
0.16460484835308187
```

In [12]:

```
#apply gridsearch

model.get_params()
```

Out[12]:

```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': None,
 'splitter': 'best'}
```

In [30]:

```
#params_grid = {'criterion':['gini','entropy'],'splitter':['best','random'],'max_depth'

params_grid = {'criterion':['gini','entropy'], 'max_depth':[4,5,6,7,8,9,10,11,12,15,20,
```

In [31]:

```
grid_search = GridSearchCV(DecisionTreeClassifier(), params_grid)
```

In [32]:

```
grid_search.fit(X_train, y_train)
```

```
C:\Users\benso\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:668: UserWa
rning: The least populated class in y has only 1 members, which is less than n_splits=5.
  % (min_groups, self.n_splits)), UserWarning)
```

Out[32]:

```
GridSearchCV(estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [4, 5, 6, 7, 8, 9, 10, 11, 12, 15, 20, 30,
                                       40, 50],
                         'random_state': [0, 1, 2, 4, 6, 8, 10, 12, 14, 16, 20,
                                          40, 42],
                         'splitter': ['best', 'random']})
```

In [34]:

```
#Using this best params (from 3.1) and the selected features (from 3.3), rebuild the De

print("best score: " + str(grid_search.best_score_))
print("test score: " + str(grid_search.score(X_test, y_test)))
decision_tree_best_params = grid_search.best_params_
print("best parameters: " + str(decision_tree_best_params))
```

```
best score: 0.1596428582289639
test score: 0.16462658984672246
best parameters: {'criterion': 'entropy', 'max_depth': 20, 'random_state': 4, 'splitte
r': 'best'}
```

In [35]:

```
#re-do with best parameters

model = DecisionTreeClassifier(criterion='gini',max_depth=20,random_state=4,splitter='b
dt_mod = model.fit(X_train, y_train)
dt_preds = dt_mod.predict(X_test)
accuracy = metrics.accuracy_score(y_test, dt_preds)
accuracy
```

```
#same as before....
```

Out[35]:

```
0.161169692357865
```

In [6]:

```
#repeat for SVM
from sklearn import svm
from sklearn.svm import LinearSVC

#from sklearn.preprocessing import MinMaxScaler
#scaling = MinMaxScaler(feature_range=(-1,1)).fit(X_train)
#X_train = scaling.transform(X_train)
#X_test = scaling.transform(X_test)

from sklearn import preprocessing
X_train = preprocessing.scale(X_train)
X_test = preprocessing.scale(X_test)

model = LinearSVC(random_state=0, tol=1e-5)
svm_mod = model.fit(X_train, y_train.ravel())

#model = svm.SVC()
#dt_mod = model.fit(X_train, y_train.ravel())
svm_preds = svm_mod.predict(X_test)
accuracy = metrics.accuracy_score(y_test, svm_preds)
accuracy
```

In [8]:

```
#repeat for LogisticRegression

from sklearn import metrics
from sklearn import preprocessing
X_train = preprocessing.scale(X_train)
X_test = preprocessing.scale(X_test)

model = LogisticRegression()
dt_mod = model.fit(X_train, y_train.ravel())
dt_preds = dt_mod.predict(X_test)
accuracy = metrics.accuracy_score(y_test, dt_preds)
accuracy
```

```
C:\Users\benso\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:765: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[8]:

```
0.053223176432220895
```

In [9]:

```
model.get_params()
```

Out[9]:

```
{'C': 1.0,
 'class_weight': None,
 'dual': False,
 'fit_intercept': True,
 'intercept_scaling': 1,
 'l1_ratio': None,
 'max_iter': 100,
 'multi_class': 'auto',
 'n_jobs': None,
 'penalty': 'l2',
 'random_state': None,
 'solver': 'lbfgs',
 'tol': 0.0001,
 'verbose': 0,
 'warm_start': False}
```

In [17]:

```
params_grid = {'C':[1.0,2.0,3.0], 'max_iter':[100,200,300]}
```

In [18]:

```
grid_search = GridSearchCV(LogisticRegression(), params_grid)
```

In [19]:

```
grid_search.fit(X_train, y_train.ravel())
```

```
C:\Users\benso\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:668: UserWa
rning: The least populated class in y has only 1 members, which is less than n_splits=5.
  % (min_groups, self.n_splits)), UserWarning)
C:\Users\benso\Anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversio
nWarning: A column-vector y was passed when a 1d array was expected. Please change the s
hape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
C:\Users\benso\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:765: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
C:\Users\benso\Anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversio
nWarning: A column-vector y was passed when a 1d array was expected. Please change the s
hape of y to (n_samples, ), for example using ravel().
  return f(*args, **kwargs)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-19-231b374c92ba> in <module>
----> 1 grid_search.fit(X_train, y_train)

~\Anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64 
     65             # extra_args > 0

~\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py in fit(self, X, y, grou
ps, **fit_params)
    839                 return results
    840 
--> 841             self._run_search(evaluate_candidates)
    842 
    843             # multimetric is determined here because in the case of a callable

~\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py in _run_search(self, ev
aluate_candidates)
   1286     def _run_search(self, evaluate_candidates):
   1287         """Search all candidates in param_grid"""
-> 1288         evaluate_candidates(ParameterGrid(self.param_grid))
   1289 
   1290 

~\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py in evaluate_candidates
(candidate_params, cv, more_results)
    807                                     (split_idx, (train, test)) in product(
    808                                     enumerate(candidate_params),
--> 809                                     enumerate(cv.split(X, y, groups))))
    810 
    811                 if len(out) < 1:

~\Anaconda3\lib\site-packages\joblib\parallel.py in __call__(self, iterable)
   1042                 self._iterating = self._original_iterator is not None
   1043 
-> 1044             while self.dispatch_one_batch(iterator):
   1045                 pass
   1046 

~\Anaconda3\lib\site-packages\joblib\parallel.py in dispatch_one_batch(self, iterator)
    857                 return False
    858             else:
--> 859                 self._dispatch(tasks)
    860                 return True
    861 

~\Anaconda3\lib\site-packages\joblib\parallel.py in _dispatch(self, batch)
    775         with self._lock:
    776             job_idx = len(self._jobs)
--> 777             job = self._backend.apply_async(batch, callback=cb)
    778             # A job can complete so quickly than its callback is
    779             # called before we get here, causing self._jobs to

~\Anaconda3\lib\site-packages\joblib\_parallel_backends.py in apply_async(self, func, ca
llback)
    206     def apply_async(self, func, callback=None):
    207         """Schedule a func to be run"""
--> 208         result = ImmediateResult(func)
    209         if callback:
    210             callback(result)

~\Anaconda3\lib\site-packages\joblib\_parallel_backends.py in __init__(self, batch)
    570         # Don't delay the application, to avoid keeping the input
    571         # arguments in memory
--> 572         self.results = batch()
    573 
    574     def get(self):

~\Anaconda3\lib\site-packages\joblib\parallel.py in __call__(self)
    261         with parallel_backend(self._backend, n_jobs=self._n_jobs):
    262             return [func(*args, **kwargs)
--> 263                     for func, args, kwargs in self.items]
    264 
    265     def __reduce__(self):

~\Anaconda3\lib\site-packages\joblib\parallel.py in <listcomp>(.0)
    261         with parallel_backend(self._backend, n_jobs=self._n_jobs):
    262             return [func(*args, **kwargs)
--> 263                     for func, args, kwargs in self.items]
    264 
    265     def __reduce__(self):
```

```
~\Anaconda3\lib\site-packages\sklearn\utils\fixes.py in __call__(self, *args, **kwargs)
    220     def __call__(self, *args, **kwargs):
    221         with config_context(**self.config):
--> 222             return self.function(*args, **kwargs)

~\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py in _fit_and_score(e
stimator, X, y, scorer, train, test, verbose, parameters, fit_params, return_train_scor
e, return_parameters, return_n_test_samples, return_times, return_estimator, split_progr
ess, candidate_progress, error_score)
    591             estimator.fit(X_train, **fit_params)
    592         else:
--> 593             estimator.fit(X_train, y_train, **fit_params)
    594 
    595     except Exception as e:

~\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py in fit(self, X, y, sampl
e_weight)
   1414                       penalty=penalty, max_squared_sum=max_squared_sum,
   1415                       sample_weight=sample_weight)
-> 1416             for class_, warm_start_coef_ in zip(classes_, warm_start_coef))
   1417 
   1418         fold_coefs_, _, n_iter_ = zip(*fold_coefs_)

~\Anaconda3\lib\site-packages\joblib\parallel.py in __call__(self, iterable)
   1039             # remaining jobs.
   1040             self._iterating = False
-> 1041             if self.dispatch_one_batch(iterator):
   1042                 self._iterating = self._original_iterator is not None
   1043 

~\Anaconda3\lib\site-packages\joblib\parallel.py in dispatch_one_batch(self, iterator)
    857                 return False
    858             else:
--> 859                 self._dispatch(tasks)
    860                 return True
    861 

~\Anaconda3\lib\site-packages\joblib\parallel.py in _dispatch(self, batch)
    775         with self._lock:
    776             job_idx = len(self._jobs)
--> 777             job = self._backend.apply_async(batch, callback=cb)
    778             # A job can complete so quickly than its callback is
    779             # called before we get here, causing self._jobs to

~\Anaconda3\lib\site-packages\joblib\_parallel_backends.py in apply_async(self, func, ca
llback)
    206     def apply_async(self, func, callback=None):
    207         """Schedule a func to be run"""
--> 208         result = ImmediateResult(func)
    209         if callback:
    210             callback(result)

~\Anaconda3\lib\site-packages\joblib\_parallel_backends.py in __init__(self, batch)
    570         # Don't delay the application, to avoid keeping the input
    571         # arguments in memory
--> 572         self.results = batch()
    573 
    574     def get(self):

~\Anaconda3\lib\site-packages\joblib\parallel.py in __call__(self)
    261         with parallel_backend(self._backend, n_jobs=self._n_jobs):
    262             return [func(*args, **kwargs)
--> 263                     for func, args, kwargs in self.items]
    264 
    265     def __reduce__(self):
```

```
~\Anaconda3\lib\site-packages\joblib\parallel.py in <listcomp>(.0)
    261         with parallel_backend(self._backend, n_jobs=self._n_jobs):
    262             return [func(*args, **kwargs)
--> 263                     for func, args, kwargs in self.items]
    264 
    265     def __reduce__(self):

~\Anaconda3\lib\site-packages\sklearn\utils\fixes.py in __call__(self, *args, **kwargs)
    220     def __call__(self, *args, **kwargs):
    221         with config_context(**self.config):
--> 222             return self.function(*args, **kwargs)

~\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py in _logistic_regression_
path(X, y, pos_class, Cs, fit_intercept, max_iter, tol, verbose, solver, coef, class_wei
ght, dual, penalty, intercept_scaling, multi_class, random_state, check_input, max_squar
ed_sum, sample_weight, l1_ratio)
    759                 func, w0, method="L-BFGS-B", jac=True,
    760                 args=(X, target, 1. / C, sample_weight),
--> 761                 options={"iprint": iprint, "gtol": tol, "maxiter": max_iter}
    762             )
    763             n_iter_i = _check_optimize_result(

~\Anaconda3\lib\site-packages\scipy\optimize\_minimize.py in minimize(fun, x0, args, met
hod, jac, hess, hessp, bounds, constraints, tol, callback, options)
    618     elif meth == 'l-bfgs-b':
    619         return _minimize_lbfgsb(fun, x0, args, jac, bounds,
--> 620                                 callback=callback, **options)
    621     elif meth == 'tnc':
    622         return _minimize_tnc(fun, x0, args, jac, bounds, callback=callback,

~\Anaconda3\lib\site-packages\scipy\optimize\lbfgsb.py in _minimize_lbfgsb(fun, x0, arg
s, jac, bounds, disp, maxcor, ftol, gtol, eps, maxfun, maxiter, iprint, callback, maxls,
finite_diff_rel_step, **unknown_options)
    358             # until the completion of the current minimization iteration.
    359             # Overwrite f and g:
--> 360             f, g = func_and_grad(x)
    361         elif task_str.startswith(b'NEW_X'):
    362             # new iteration

~\Anaconda3\lib\site-packages\scipy\optimize\_differentiable_functions.py in fun_and_gra
d(self, x)
    258         if not np.array_equal(x, self.x):
    259             self._update_x_impl(x)
--> 260         self._update_fun()
    261         self._update_grad()
    262         return self.f, self.g

~\Anaconda3\lib\site-packages\scipy\optimize\_differentiable_functions.py in _update_fun
(self)
    224     def _update_fun(self):
    225         if not self.f_updated:
--> 226             self._update_fun_impl()
    227             self.f_updated = True
    228 

~\Anaconda3\lib\site-packages\scipy\optimize\_differentiable_functions.py in update_fun
()
    131 
    132         def update_fun():
--> 133             self.f = fun_wrapped(self.x)
    134 
    135         self._update_fun_impl = update_fun

~\Anaconda3\lib\site-packages\scipy\optimize\_differentiable_functions.py in fun_wrapped
```

```
(x)
    128         def fun_wrapped(x):
    129             self.nfev += 1
--> 130             return fun(x, *args)
    131 
    132         def update_fun():

~\Anaconda3\lib\site-packages\scipy\optimize\optimize.py in __call__(self, x, *args)
     72     def __call__(self, x, *args):
     73         """ returns the the function value """
---> 74         self._compute_if_needed(x, *args)
     75         return self._value
     76 

~\Anaconda3\lib\site-packages\scipy\optimize\optimize.py in _compute_if_needed(self, x,
 *args)
     66         if not np.all(x == self.x) or self._value is None or self.jac is None:
     67             self.x = np.asarray(x).copy()
---> 68             fg = self.fun(x, *args)
     69             self.jac = fg[1]
     70             self._value = fg[0]

~\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py in func(x, *args)
    734         target = Y_multi
    735         if solver == 'lbfgs':
--> 736             def func(x, *args): return _multinomial_loss_grad(x, *args)[0:2]
    737         elif solver == 'newton-cg':
    738             def func(x, *args): return _multinomial_loss(x, *args)[0]

~\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py in _multinomial_loss_gra
d(w, X, Y, alpha, sample_weight)
    346     grad = np.zeros((n_classes, n_features + bool(fit_intercept)),
    347                     dtype=X.dtype)
--> 348     loss, p, w = _multinomial_loss(w, X, Y, alpha, sample_weight)
    349     sample_weight = sample_weight[:, np.newaxis]
    350     diff = sample_weight * (p - Y)

~\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py in _multinomial_loss(w,
 X, Y, alpha, sample_weight)
    297     p += intercept
    298     p -= logsumexp(p, axis=1)[:, np.newaxis]
--> 299     loss = -(sample_weight * Y * p).sum()
    300     loss += 0.5 * alpha * squared_norm(w)
    301     p = np.exp(p, p)

~\Anaconda3\lib\site-packages\numpy\core\_methods.py in _sum(a, axis, dtype, out, keepdi
ms, initial, where)
     45 def _sum(a, axis=None, dtype=None, out=None, keepdims=False,
     46          initial=_NoValue, where=True):
---> 47     return umr_sum(a, axis, dtype, out, keepdims, initial, where)
     48 
     49 def _prod(a, axis=None, dtype=None, out=None, keepdims=False,

KeyboardInterrupt: 
```

In [7]:

```python
#repeat for KNeighborsClassifier

from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics


model = KNeighborsClassifier()

k_neigh = model.fit(X_train, y_train)
```

```
k_preds = k_neigh.predict(X_test)
accuracy = metrics.accuracy_score(y_test, k_preds)
accuracy
```

```
C:\Users\benso\Anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:179: Dat
aConversionWarning: A column-vector y was passed when a 1d array was expected. Please ch
ange the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
```

Out[7]:

```
0.11312099141211002
```

In [8]:

```
model.get_params()
```

Out[8]:

```
{'algorithm': 'auto',
 'leaf_size': 30,
 'metric': 'minkowski',
 'metric_params': None,
 'n_jobs': None,
 'n_neighbors': 5,
 'p': 2,
 'weights': 'uniform'}
```

In [9]:

```
params_grid = {'algorithm':['auto','ball_tree','kd_tree'], 'n_neighbors':[2,5,10,20]}
```

In [10]:

```
grid_search = GridSearchCV(KNeighborsClassifier(), params_grid)
```

In [11]:

```
grid_search.fit(X_train, y_train.ravel())
```

```
C:\Users\benso\Anaconda3\lib\site-packages\sklearn\model_selection\_split.py:668: UserWa
rning: The least populated class in y has only 1 members, which is less than n_splits=5.
  % (min_groups, self.n_splits)), UserWarning)
```

Out[11]:

```
GridSearchCV(estimator=KNeighborsClassifier(),
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree'],
                         'n_neighbors': [2, 5, 10, 20]})
```

In [12]:

```
print("best score: " + str(grid_search.best_score_))
print("test score: " + str(grid_search.score(X_test, y_test)))
K_neighbors_best_params = grid_search.best_params_
print("best parameters: " + str(K_neighbors_best_params))
```

```
best score: 0.12534873906768934
test score: 0.12947059462985108
best parameters: {'algorithm': 'auto', 'n_neighbors': 20}
```

In [ ]:

```
#two of the classifiers took more than 3 hours to run, so the data is inconclusive. Bas
#'auto' algorithm, with 20 neighbors.
```