Project IV

Telehealth - Fall Detection

Caner Canlıer 21702121

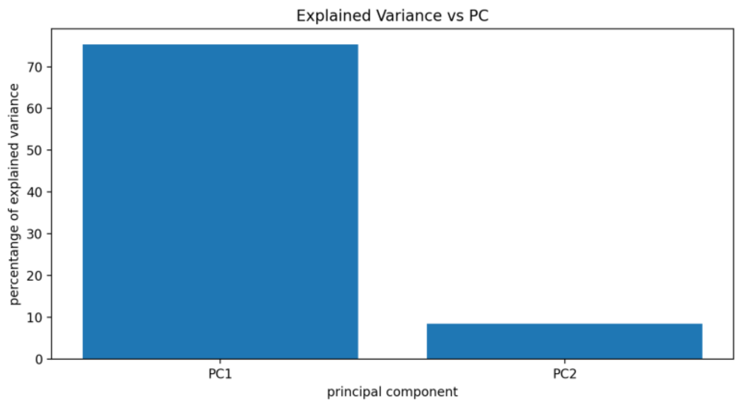
Bilkent University

GE 461: Introduction to Data Science

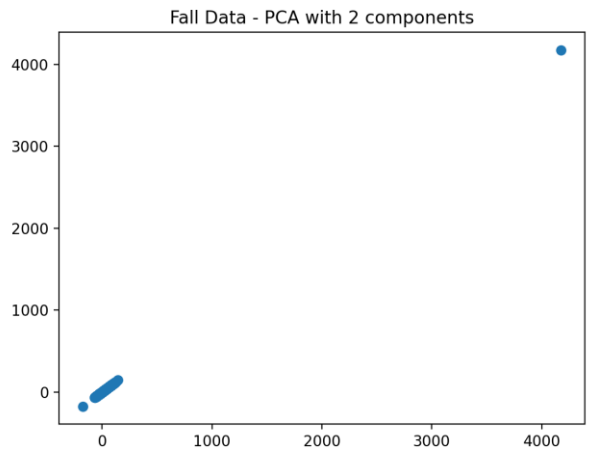
5th of May, 2022

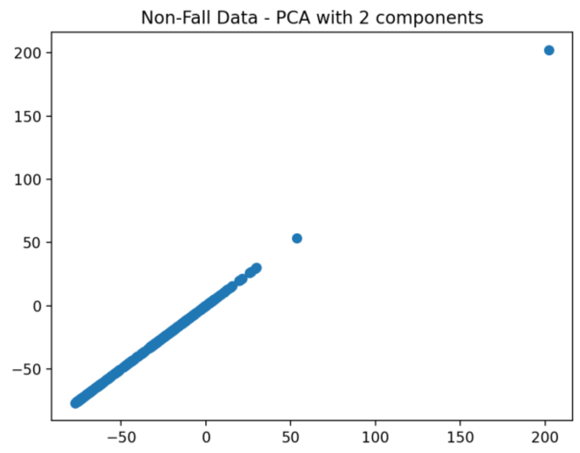
Part A

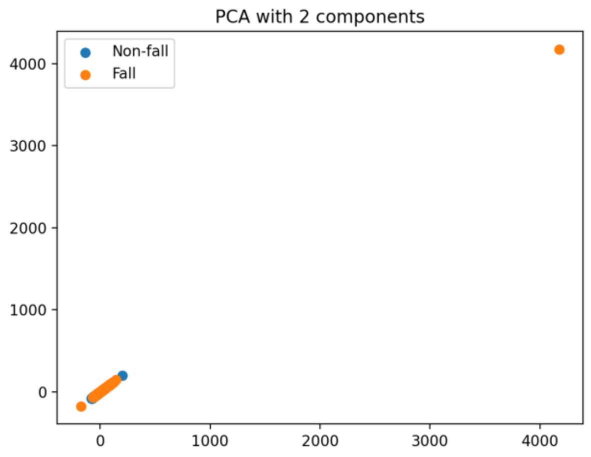
First, I use the projections of 566 data samples onto the first and second PCs and calculate the total variance. First principal component and second principal component were provided 75.3% and 8.5% variance, respectively. Total variance that we capture was 83.8%. It can be seen from the below bar graph.



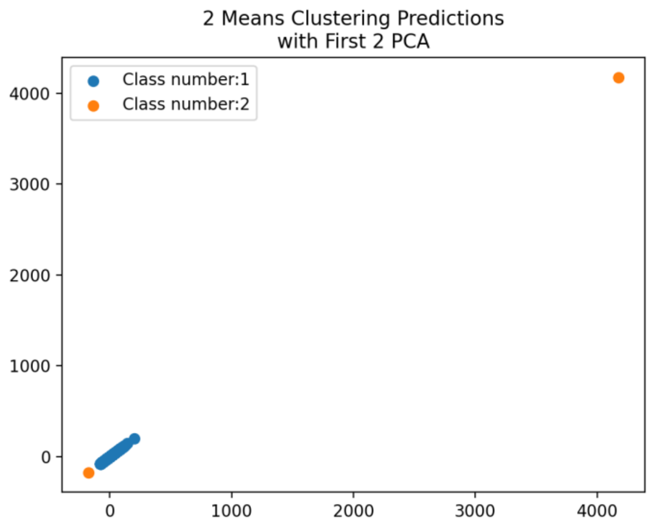
Second, I project the given input to first two principal components.

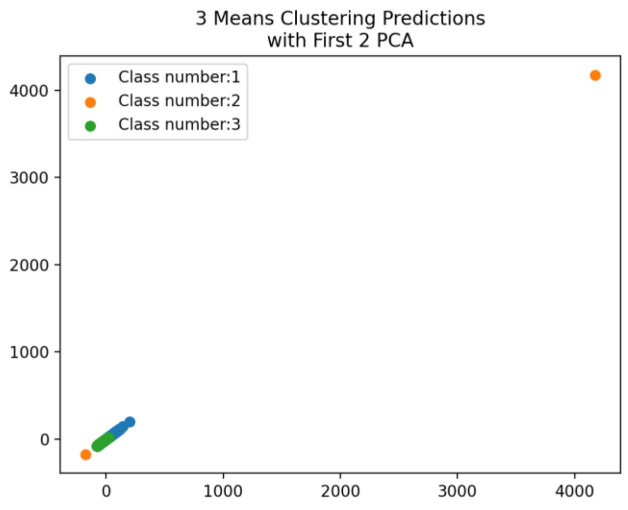


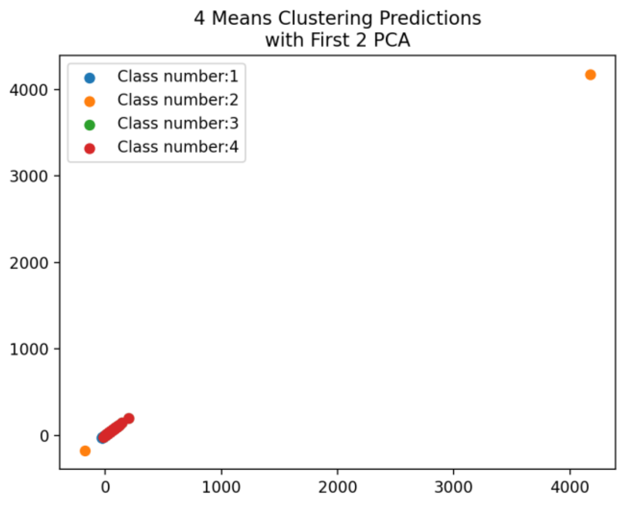


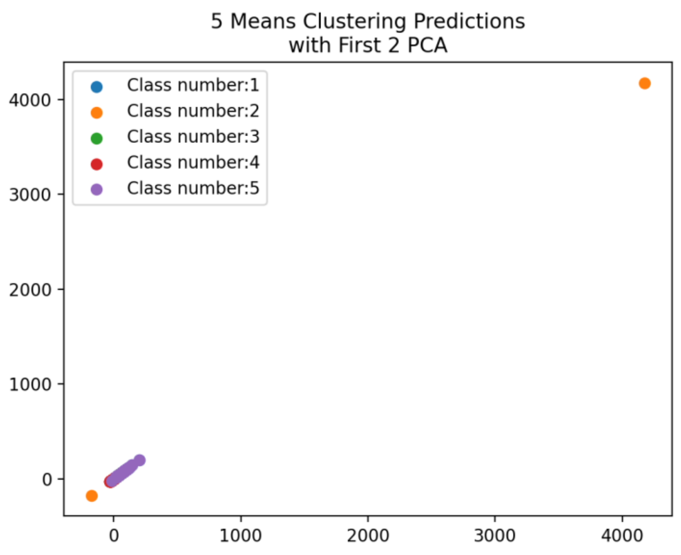


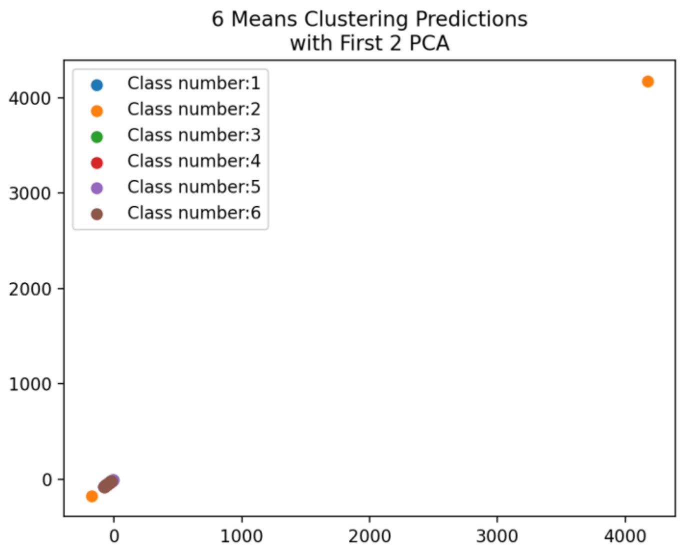
From the above graph we can observe an outlier in fall data set. I run k-means clustering to separate data into clusters. I try different values for k value such as 2,3,4,5,6. The graphs of K-means clustering predictions of top 2 PCA projections with different k can be seen below.











From the graphs, it can be seen that there is an outlier which makes harder us to see the clusters. Also, after class number 3, all clusters start to overlap with each other and separation becomes unclear.

I took the clusters obtained when N=2 and calculate the overlap/consistency accuracy. According to my calculation, 55.1% of our predictions were true. Basically, the algorithm clusters all of the data together. This is a slight improvement over random assignment, which is ineffective. I wonder the reason why 2-mean clustering doesn't perform well and I thought that if we remove the outlier point, we might get better solution. So, I remove it and repeat the same procedure again to calculate overlap accuracy. After removing the outlier, our accuracy rate increased from 55.1% to 78.4%.

As a result, since this is an unsupervised learning, we don't expect perform well as much as supervised learning algorithms. However, after removing the outliers we achieve 78.4% accuracy rate which is relatively good. Although it is not reliable as much as MLP or SVM, fall detection is possible based on these measurements.

Part B

For this part, I split the all data into 3 parts which are train(70%), test(15%) and validation(15%) by using train_test_split function from sklearn.model_selection. After that I build 2 models. First model is a support-vector-machine (SVM) classifier that detects the action label (fall or non-fall) with high accuracy. Then I build a multi-layer perceptron (MLP) classifier. For bot classifier I experiment with different hyperparameters to maximize the classification accuracy. To calculate classification accuracy, I check confusion matrix. After deciding the best parameters, I achieve a final classification accuracy by using my test data set. See the details of both classifiers:

SVM

To model a SVM model, I used SVC function from sklearn.svm. SVC function can be seen in the below:

class sklearn.svm.SVC(*, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=- 1, decision_function_shape='ovr', break_ties=False, random_state=None)

To try different combinations, I change some of those hyperparameters. I observe different cases when we change the kernel type, gamma and regularization parameter.

- **kernel_opt=["linear", "poly","rbf", "sigmoid"]**
- **c_opt=[0.01,0.1,1,10,100]**
- **gamma_opt=["scale","auto"]**

After trying all those values, the model come up with best parameters which provides 100% accuracy for validation set. Best parameters for this model was:

- Kernel = “poly”, C=0.01, gamma="auto"

I trained another SVM model with this parameters and test on test data set and come up with the same accuracy result which is 100%.

In addition to that, all the intermediate results can be seen below:

poly auto 0.01	**1**
poly auto 0.1	1
poly auto 1	1
poly auto 10	1
poly auto 100	1
linear scale 0.01	0.988235
linear scale 0.1	0.988235
linear scale 1	0.988235
linear scale 10	0.988235
linear scale 100	0.988235
linear auto 0.01	0.988235
linear auto 0.1	0.988235
linear auto 1	0.988235
linear auto 10	0.988235
linear auto 100	0.988235
rbf scale 100	0.988235
rbf scale 1	0.976471

rbf scale 10	0.976471
poly scale 100	0.964706
sigmoid scale 1	0.964706
sigmoid scale 10	0.941176
poly scale 10	0.917647
rbf scale 0.1	0.917647
rbf auto 1	0.905882
rbf auto 10	0.905882
rbf auto 100	0.905882
poly scale 1	0.882353
sigmoid scale 0.1	0.870588
sigmoid scale 100	0.858824
rbf auto 0.1	0.823529
poly scale 0.01	0.588235
poly scale 0.1	0.588235
rbf scale 0.01	0.588235
rbf auto 0.01	0.588235
sigmoid scale 0.01	0.588235
sigmoid auto 0.01	0.588235
sigmoid auto 0.1	0.588235
sigmoid auto 1	0.588235
sigmoid auto 10	0.141176
sigmoid auto 100	0.141176

From this table, we can interpret that using sigmoid as activation function generally doesn't perform well while all combinations with linear activation function performs approximately 99% accuracy. Also, while poly auto combination gives the best result, poly scale combination doesn't perform that well.

MLP

To model a MLP model, I used MLPClassifier function from sklearn.neural_network. MLPClassifier function can be seen in the below:

class sklearn.neural_network.MLPClassifier(hidden_layer_sizes=(100,), activation='relu', *, solver='adam', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True, random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000)

To try different combinations, I change some of those hyperparameters. I observe different cases when we change the hidden layer size, activation function, solver, learning rate schedule and learning rate.

- **hidden_layer_size= [(15,),(15,15),(30,30)]**
- **activation= ["identity","logistic", "tanh", "relu"]**
- **solver=["lbfgs", "sgd",]**
- **learning_rate=["constant", "invscaling", "adaptive"]**
- **learning_rate_init=[0.001,0.01,0.1]**

After trying all those values, the model come up with best parameters which provides 100% accuracy for validation set. Best parameters for this model was:

- hidden_layer_sizes=(15,), activation="identity", solver="sgd", learning_rate="constant", learning_rate_init=0.001

I trained another MLP model with this parameters and test on test data set and come up with the same accuracy result which is 100%.

In addition to that, all the intermediate results can be seen below:

(15,) identity sgd constant 0.001	**1**
(15,) identity sgd adaptive 0.001	1
(15,) logistic sgd constant 0.001	1
(15,) logistic sgd adaptive 0.001	1
(15,) relu lbfgs constant 0.001	1
(15,) relu lbfgs constant 0.01	1
(15,) relu lbfgs constant 0.1	1
(15,) relu lbfgs invscaling 0.001	1
(15,) relu lbfgs invscaling 0.01	1
(15,) relu lbfgs invscaling 0.1	1
(15,) relu lbfgs adaptive 0.001	1
(15,) relu lbfgs adaptive 0.01	1
(15,) relu lbfgs adaptive 0.1	1
(15, 15) logistic lbfgs constant 0.001	1
(15, 15) logistic lbfgs constant 0.01	1
(15, 15) logistic lbfgs constant 0.1	1
(15, 15) logistic lbfgs invscaling 0.001	1
(15, 15) logistic lbfgs invscaling 0.01	1
(15, 15) logistic lbfgs invscaling 0.1	1
(15, 15) logistic lbfgs adaptive 0.001	1
(15, 15) logistic lbfgs adaptive 0.01	1
(15, 15) logistic lbfgs adaptive 0.1	1
(15, 15) tanh sgd constant 0.001	1
(15, 15) tanh sgd constant 0.01	1
(15, 15) tanh sgd adaptive 0.001	1
(15, 15) tanh sgd adaptive 0.01	1
(15,) identity lbfgs constant 0.001	0.988235
(15,) identity lbfgs constant 0.01	0.988235
(15,) identity lbfgs constant 0.1	0.988235
(15,) identity lbfgs invscaling 0.001	0.988235
(15,) identity lbfgs invscaling 0.01	0.988235
(15,) identity lbfgs invscaling 0.1	0.988235
(15,) identity lbfgs adaptive 0.001	0.988235
(15,) identity lbfgs adaptive 0.01	0.988235
(15,) identity lbfgs adaptive 0.1	0.988235

(15,) identity sgd constant 0.01	0.988235
(15,) identity sgd adaptive 0.01	0.988235
(15,) logistic sgd constant 0.01	0.988235
(15,) logistic sgd constant 0.1	0.988235
(15,) logistic sgd adaptive 0.01	0.988235
(15,) logistic sgd adaptive 0.1	0.988235
(15,) tanh lbfgs constant 0.001	0.988235
(15,) tanh lbfgs constant 0.01	0.988235
(15,) tanh lbfgs constant 0.1	0.988235
(15,) tanh lbfgs invscaling 0.001	0.988235
(15,) tanh lbfgs invscaling 0.01	0.988235
(15,) tanh lbfgs invscaling 0.1	0.988235
(15,) tanh lbfgs adaptive 0.001	0.988235
(15,) tanh lbfgs adaptive 0.01	0.988235
(15,) tanh lbfgs adaptive 0.1	0.988235
(15,) tanh sgd constant 0.001	0.988235
(15,) tanh sgd constant 0.01	0.988235
(15,) tanh sgd constant 0.1	0.988235
(15,) tanh sgd adaptive 0.001	0.988235
(15,) tanh sgd adaptive 0.01	0.988235
(15,) tanh sgd adaptive 0.1	0.988235
(15,) relu sgd constant 0.001	0.988235
(15,) relu sgd constant 0.01	0.988235
(15,) relu sgd adaptive 0.001	0.988235
(15,) relu sgd adaptive 0.01	0.988235
(15, 15) identity lbfgs constant 0.001	0.988235
(15, 15) identity lbfgs constant 0.01	0.988235
(15, 15) identity lbfgs constant 0.1	0.988235
(15, 15) identity lbfgs invscaling 0.001	0.988235
(15, 15) identity lbfgs invscaling 0.01	0.988235
(15, 15) identity lbfgs invscaling 0.1	0.988235
(15, 15) identity lbfgs adaptive 0.001	0.988235
(15, 15) identity lbfgs adaptive 0.01	0.988235
(15, 15) identity lbfgs adaptive 0.1	0.988235
(15, 15) identity sgd constant 0.001	0.988235
(15, 15) identity sgd adaptive 0.001	0.988235
(15, 15) logistic sgd constant 0.001	0.988235
(15, 15) logistic sgd constant 0.01	0.988235
(15, 15) logistic sgd constant 0.1	0.988235
(15, 15) logistic sgd adaptive 0.001	0.988235
(15, 15) logistic sgd adaptive 0.01	0.988235
(15, 15) logistic sgd adaptive 0.1	0.988235
(15, 15) tanh sgd constant 0.1	0.988235

(15, 15) tanh sgd invscaling 0.1	0.988235
(15, 15) tanh sgd adaptive 0.1	0.988235
(15, 15) relu lbfgs constant 0.001	0.988235
(15, 15) relu lbfgs constant 0.01	0.988235
(15, 15) relu lbfgs constant 0.1	0.988235
(15, 15) relu lbfgs invscaling 0.001	0.988235
(15, 15) relu lbfgs invscaling 0.01	0.988235
(15, 15) relu lbfgs invscaling 0.1	0.988235
(15, 15) relu lbfgs adaptive 0.001	0.988235
(15, 15) relu lbfgs adaptive 0.01	0.988235
(15, 15) relu lbfgs adaptive 0.1	0.988235
(30, 30) identity lbfgs constant 0.001	0.988235
(30, 30) identity lbfgs constant 0.01	0.988235
(30, 30) identity lbfgs constant 0.1	0.988235
(30, 30) identity lbfgs invscaling 0.001	0.988235
(30, 30) identity lbfgs invscaling 0.01	0.988235
(30, 30) identity lbfgs invscaling 0.1	0.988235
(30, 30) identity lbfgs adaptive 0.001	0.988235
(30, 30) identity lbfgs adaptive 0.01	0.988235
(30, 30) identity lbfgs adaptive 0.1	0.988235
(30, 30) identity sgd constant 0.001	0.988235
(30, 30) identity sgd adaptive 0.001	0.988235
(30, 30) logistic lbfgs constant 0.001	0.988235
(30, 30) logistic lbfgs constant 0.01	0.988235
(30, 30) logistic lbfgs constant 0.1	0.988235
(30, 30) logistic lbfgs invscaling 0.001	0.988235
(30, 30) logistic lbfgs invscaling 0.01	0.988235
(30, 30) logistic lbfgs invscaling 0.1	0.988235
(30, 30) logistic lbfgs adaptive 0.001	0.988235
(30, 30) logistic lbfgs adaptive 0.01	0.988235
(30, 30) logistic lbfgs adaptive 0.1	0.988235
(30, 30) logistic sgd constant 0.001	0.988235
(30, 30) logistic sgd constant 0.01	0.988235
(30, 30) logistic sgd constant 0.1	0.988235
(30, 30) logistic sgd adaptive 0.001	0.988235
(30, 30) logistic sgd adaptive 0.01	0.988235
(30, 30) logistic sgd adaptive 0.1	0.988235
(30, 30) tanh sgd constant 0.01	0.988235
(30, 30) tanh sgd adaptive 0.01	0.988235
(30, 30) tanh sgd adaptive 0.1	0.988235
(30, 30) relu lbfgs constant 0.001	0.988235
(30, 30) relu lbfgs constant 0.01	0.988235
(30, 30) relu lbfgs constant 0.1	0.988235

(30, 30) relu lbfgs invscaling 0.001	0.988235
(30, 30) relu lbfgs invscaling 0.01	0.988235
(30, 30) relu lbfgs invscaling 0.1	0.988235
(30, 30) relu lbfgs adaptive 0.001	0.988235
(30, 30) relu lbfgs adaptive 0.01	0.988235
(30, 30) relu lbfgs adaptive 0.1	0.988235
(30, 30) relu sgd constant 0.01	0.988235
(30, 30) relu sgd adaptive 0.01	0.988235
(15,) identity sgd invscaling 0.1	0.976471
(15,) identity sgd adaptive 0.1	0.976471
(15,) logistic lbfgs constant 0.001	0.976471
(15,) logistic lbfgs constant 0.01	0.976471
(15,) logistic lbfgs constant 0.1	0.976471
(15,) logistic lbfgs invscaling 0.001	0.976471
(15,) logistic lbfgs invscaling 0.01	0.976471
(15,) logistic lbfgs invscaling 0.1	0.976471
(15,) logistic lbfgs adaptive 0.001	0.976471
(15,) logistic lbfgs adaptive 0.01	0.976471
(15,) logistic lbfgs adaptive 0.1	0.976471
(15,) tanh sgd invscaling 0.01	0.976471
(15, 15) identity sgd invscaling 0.01	0.976471
(15, 15) tanh lbfgs constant 0.001	0.976471
(15, 15) tanh lbfgs constant 0.01	0.976471
(15, 15) tanh lbfgs constant 0.1	0.976471
(15, 15) tanh lbfgs invscaling 0.001	0.976471
(15, 15) tanh lbfgs invscaling 0.01	0.976471
(15, 15) tanh lbfgs invscaling 0.1	0.976471
(15, 15) tanh lbfgs adaptive 0.001	0.976471
(15, 15) tanh lbfgs adaptive 0.01	0.976471
(15, 15) tanh lbfgs adaptive 0.1	0.976471
(15, 15) relu sgd constant 0.001	0.976471
(15, 15) relu sgd constant 0.01	0.976471
(15, 15) relu sgd adaptive 0.001	0.976471
(15, 15) relu sgd adaptive 0.01	0.976471
(30, 30) identity sgd invscaling 0.01	0.976471
(30, 30) tanh lbfgs constant 0.001	0.976471
(30, 30) tanh lbfgs constant 0.01	0.976471
(30, 30) tanh lbfgs constant 0.1	0.976471
(30, 30) tanh lbfgs invscaling 0.001	0.976471
(30, 30) tanh lbfgs invscaling 0.01	0.976471
(30, 30) tanh lbfgs invscaling 0.1	0.976471
(30, 30) tanh lbfgs adaptive 0.001	0.976471
(30, 30) tanh lbfgs adaptive 0.01	0.976471

(30, 30) tanh lbfgs adaptive 0.1	0.976471
(30, 30) tanh sgd constant 0.001	0.976471
(30, 30) tanh sgd constant 0.1	0.976471
(30, 30) tanh sgd invscaling 0.1	0.976471
(30, 30) tanh sgd adaptive 0.001	0.976471
(30, 30) relu sgd constant 0.001	0.976471
(30, 30) relu sgd adaptive 0.001	0.976471
(15,) identity sgd invscaling 0.01	0.964706
(15,) logistic sgd invscaling 0.1	0.964706
(30, 30) identity sgd invscaling 0.001	0.964706
(15,) relu sgd invscaling 0.001	0.952941
(30, 30) logistic sgd invscaling 0.1	0.952941
(15,) identity sgd invscaling 0.001	0.929412
(15,) logistic sgd invscaling 0.01	0.929412
(15, 15) tanh sgd invscaling 0.01	0.929412
(30, 30) relu sgd invscaling 0.001	0.917647
(30, 30) tanh sgd invscaling 0.01	0.905882
(15,) relu sgd invscaling 0.1	0.894118
(15,) relu sgd invscaling 0.01	0.882353
(15, 15) relu sgd invscaling 0.001	0.870588
(30, 30) tanh sgd invscaling 0.001	0.835294
(15,) tanh sgd invscaling 0.1	0.811765
(15, 15) relu sgd invscaling 0.01	0.776471
(15,) tanh sgd invscaling 0.001	0.647059
(15,) relu sgd constant 0.1	0.6
(15,) relu sgd adaptive 0.1	0.6
(30, 30) relu sgd invscaling 0.01	0.6
(15,) identity sgd constant 0.1	0.588235
(15, 15) identity sgd constant 0.01	0.588235
(15, 15) identity sgd constant 0.1	0.588235
(15, 15) identity sgd invscaling 0.001	0.588235
(15, 15) identity sgd invscaling 0.1	0.588235
(15, 15) identity sgd adaptive 0.01	0.588235
(15, 15) identity sgd adaptive 0.1	0.588235
(15, 15) logistic sgd invscaling 0.1	0.588235
(15, 15) relu sgd constant 0.1	0.588235
(15, 15) relu sgd adaptive 0.1	0.588235
(30, 30) identity sgd constant 0.01	0.588235
(30, 30) identity sgd constant 0.1	0.588235
(30, 30) identity sgd 0.1	0.588235
(30, 30) identity sgd adaptive 0.01	0.588235
(30, 30) identity sgd adaptive 0.1	0.588235
(30, 30) logistic sgd invscaling 0.001	0.588235

(30, 30) logistic sgd invscaling 0.01	0.588235
(30, 30) relu sgd constant 0.1	0.588235
(30, 30) relu sgd adaptive 0.1	0.588235
(15, 15) tanh sgd invscaling 0.001	0.576471
(15,) logistic sgd invscaling 0.001	0.552941
(15, 15) logistic sgd invscaling 0.001	0.411765
(15, 15) logistic sgd invscaling 0.01	0.411765
(15, 15) relu sgd invscaling 0.1	0.411765
(30, 30) relu sgd invscaling 0.1	0.411765

From the table, we can interpret that models with less hidden layers tend to perform better. Also choosing invscaling as learning rate schedule performs worse when we compare with others. Activation function and solver seem like less influential when we perform fall detection.

Comparison of MLP and SVM

For both SVM and MLP models, we obtained 100% accuracy for both validation and test data set. It is hard to compare those two models since their accuracy scores for different parameters are similar and they both give perfect accuracy for some of parameters. However, after doing further analysis we can say that MLP performs better than SVM since the chance of getting above 90% accuracy with different variables with MLP is higher than with SVM. On the other hand, SVM model might be better to option to make faster decisions since it performs faster than the MLP.