DS-203 Assignment 8

Chirag Garg
190100042
Electrical Engineering, Dual Degree

Ans 1.)

Question 1			
Part No.	**Input Variables**	**Output Variables**	**Type of Problem**
a.)	Grades of 50 courses	5 numbers (mentioned in the question)	PCA
b.)	Students' Performance in courses and extra curriculars	Personality group of students	Clustering
c.)	Performance; 1st job type; extra curricular activities	Salary after 2 years	Regression
d.)	Performance in courses and extra curricular activities	Best job type for person	Classification

Ans 2.)

Question 2

Problem	Framework	Target output variable	Parameters	Hyperparameters and their typical value range	Scikit-learn commands for their defining training and testing
Classification	SVM-C with Gaussian kernel	OHE	Support Vectors, their coefficients and intercepts	C(regularisation)**[0.5-2]**, gamma(kernel coefficient)**['scale'/'auto']**, tol(tolerance)**[10^-5 - 10^-3]**	svmClass = sklearn.svm.SVC(); svmClass.fit(x, y); svmClass.predict(X_)
	NN with one hidden layer	OHE	Weights and biases	hidden_layer_size**[20-1000]**, activation**['relu'/'tanh'/'sigmoid']**, solver**['sgd'/'adam']**, alpha**[0.001(default)-1]**	mlpClass = MLPClassifier(); mlpClass.fit(x, y); mlpClass.predict(X_)
	Random Forest	OHE	No. of decision trees, No. of features considered by each tree when splitting a node	n_estimators**[10-1000]**, orientation**['gini'/'entropy']**, max_depth**[2-10]**, bootstrap**[True/False]**	rfc =sklearn.ensemble.RandomForestClassifier(); rfc.fit(x, y); rfc.predict(X_)
Regression	SVM-R with Gaussian kernel	Float	Support Vectors, their coefficients and intercepts	C(regularisation)**[0.5-2]**, gamma(kernel coefficient)**['scale'/'auto']**, tol(tolerance)**[10^-5 - 10^-3]**, epsilon**[0.01-0.1]**	svmReg = sklearn.svm.SVR(); svmReg.fit(x, y); svmReg.predict(X_)
	NN with one hidden layer	Float	Weights and biases	hidden_layer_size**[20-1000]**, activation**['relu'/'tanh'/'sigmoid']**, solver**['sgd'/'adam']**, alpha**[0.001(default)-1]**	mlpReg = MLPClassifier(); mlpReg.fit(x, y); mlpReg.predict(X_)
	Random Forest	Float	No. of decision trees, No. of features considered by each tree when splitting a node	n_estimators**[10-1000]**, orientation**['gini'/'entropy']**, max_depth**[2-10]**, bootstrap**[True/False]**	rfr =sklearn.ensemble.RandomForestRegressor(); rfr.fit(x, y); rfr.predict(X_)
Clustering	k-means	Integer	Cluster Centers	n_clusters**[4-12]**, init**['k-means++'/'random']**, max_iter**[100-1000]**, tol(tolerance)**[10^-5 - 10^-3]**	kmeansClass = sklearn.cluster.Kmeans(); kmeansClass.fit(x); kmeansClass.predict(X_)
	DBSCAN	Integer	Core Points	eps**[0.1-0.9]**, min_samples**[default: 5]**, algorithm**['auto'/'kd-tree'/'brute']**	dbs = sklearn.cluster.DBSCAN(); dbs.fit(x); dbs.predict(X_)
Dimension Reduction	PCA	None	Vectors along which data is to be projected	n_components**[1-no. of features]**, copy**[True/False]**, svd-solver**['auto'/'full'/'randomized']**	pca = sklearn.decomposition.PCA(); pca.fit(X); pca.fit_transform(X_)
	Kernel PCA	None	Vectors along which data is to be projected	n_components**[1-no. of features]**, gamma**[~1/no. of features]**, degree**[2-4]**	kpca = sklearn.decomposition.KernelPCA(); kpca.fit(X); kpca.fit_transform(X_)