```python
import pandas as pd
import numpy as np
from pandas import DataFrame
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import VotingClassifier, BaggingClassifier, BaggingRegressor, RandomForestClassifier, ExtraTreesClassifier
from sklearn.feature_selection import SelectPercentile
from sklearn.preprocessing import RobustScaler, KBinsDiscretizer, StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import cross_val_score, KFold
from sklearn.datasets import load_breast_cancer
```

Voting Classifiers

```python
#dataset
cancer=load_breast_cancer()
cancer_data   =cancer.data
cancer_target =cancer.target

#classifiers
lr = LogisticRegression()
dt = DecisionTreeClassifier()
svm= SVC(probability=True)
knn= KNeighborsClassifier()
hard_voting = VotingClassifier(estimators=[('lr', lr), ('dt', dt), ('svc', svm),
                                           ('knn',knn)],voting='hard')
soft_voting = VotingClassifier(estimators=[('lr', lr), ('dt', dt), ('svc', svm),
                                           ('knn',knn)],voting='soft')
#data preprocessing
anova=SelectPercentile(percentile=30)
scale=RobustScaler()
ct=ColumnTransformer([('scale',scale,list(range(30)))])

cancer_data=ct.fit_transform(cancer_data)
cancer_data=anova.fit_transform(cancer_data,cancer_target)

#comparison
estimators = [lr,dt,svm,knn,hard_voting,soft_voting]
for x in estimators:
    result=cross_val_score(x,cancer_data,cancer_target,cv=5)
    final_result=np.mean(result)
    print(x,final_result)
```

```
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False) 0.9437820214252446
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform') 0.9455519329296692
VotingClassifier(estimators=[('lr',
                              LogisticRegression(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)),
                             ('dt',
                              DecisionTreeClassifier(ccp_alpha=0.0,
                                                     class_weight=None,
                                                     criterion='gini',...
                                  decision_function_shape='ovr', degree=3,
                                  gamma='scale', kernel='rbf', max_iter=-1,
                                  probability=True, random_state=None,
                                  shrinking=True, tol=0.001, verbose=False)),
                             ('knn',
                              KNeighborsClassifier(algorithm='auto',
                                                   leaf_size=30,
                                                   metric='minkowski',
                                                   metric_params=None,
                                                   n_jobs=None, n_neighbors=5,
                                                   p=2, weights='uniform'))],
                 flatten_transform=True, n_jobs=None, voting='hard',
                 weights=None) 0.9525694767893185
VotingClassifier(estimators=[('lr',
                              LogisticRegression(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)),
                             ('dt',
                              DecisionTreeClassifier(ccp_alpha=0.0,
                                                     class_weight=None,
                                                     criterion='gini',...
                                  decision_function_shape='ovr', degree=3,
                                  gamma='scale', kernel='rbf', max_iter=-1,
                                  probability=True, random_state=None,
                                  shrinking=True, tol=0.001, verbose=False)),
                             ('knn',
                              KNeighborsClassifier(algorithm='auto',
                                                   leaf_size=30,
                                                   metric='minkowski',
                                                   metric_params=None,
                                                   n_jobs=None, n_neighbors=5,
                                                   p=2, weights='uniform'))],
                 flatten_transform=True, n_jobs=None, voting='soft',
                 weights=None) 0.947290793355069
```

Bagging and Pasting

```
kf = KFold(n_splits=3)
bagging = BaggingClassifier(SVC(), n_estimators=500,
                            max_samples=0.8, bootstrap=True, n_jobs=-1)

bagging_result = cross_val_score(bagging,cancer_data,cancer_target,cv=kf,n_jobs=-1)
print("bagging results",bagging_result)
print("average of bagging:",np.mean(bagging_result))
print("********************************************")

pasting = BaggingClassifier(SVC(), n_estimators=500,
                            max_samples=0.8, bootstrap=False, n_jobs=-1)

pasting_result = cross_val_score(pasting,cancer_data,cancer_target,cv=kf,n_jobs=-1)
print("pasting results",pasting_result)
print("average of pasting:",np.mean(pasting_result))
print("********************************************")
```

```
bagging results [0.89473684 0.97368421 0.96296296]
average of bagging: 0.9437946718648473
********************************************
pasting results [0.89473684 0.97368421 0.96296296]
average of pasting: 0.9437946718648473
********************************************
```

With Regressors

```
#dataset
carset=pd.read_csv('/content/CarPrice_Assignment.csv')
carset=carset.drop(['car_ID','symboling','CarName','doornumber','carbody','enginelocation'],axis=1)
car_target=carset['price']
car_data=carset.iloc[:,0:19]
#data preprocessing
kf = KFold(n_splits=4)
bins = KBinsDiscretizer(n_bins=5, encode='onehot-dense', strategy='uniform')
numeric_cols=car_data.select_dtypes(include=np.number).columns
print("numeric_cols",numeric_cols)
numeric_cols=numeric_cols.delete([1,2,3])
categorical_cols=car_data.select_dtypes(exclude=np.number).columns
print("categorical_cols",categorical_cols)

ct=ColumnTransformer([('scaling',StandardScaler(),numeric_cols),
                      ('binning',bins,['carheight','carwidth','carlength']),
                      ('categorical',OneHotEncoder(sparse=False,handle_unknown='ignore'),categorical_cols)
                      ])

final_data=ct.fit_transform(car_data)

#bagging_regressor
bagging_regressor=BaggingRegressor(DecisionTreeRegressor(), n_estimators=250,
                                   max_samples=0.8,max_features=0.4, bootstrap=True, n_jobs=-1)


bagging_regressor_result = cross_val_score(bagging_regressor,final_data,car_target,cv=kf,n_jobs=-1)
print("bagging results",bagging_regressor_result)
print("average of bagging:",np.mean(bagging_regressor_result))
print("********************************************")
```

```
numeric_cols Index(['wheelbase', 'carlength', 'carwidth', 'carheight', 'curbweight',
       'enginesize', 'boreratio', 'stroke', 'compressionratio', 'horsepower',
       'peakrpm', 'citympg', 'highwaympg'],
      dtype='object')
categorical_cols Index(['fueltype', 'aspiration', 'drivewheel', 'enginetype', 'cylindernumber',
       'fuelsystem'],
      dtype='object')
bagging results [0.8848483  0.82759289 0.75087733 0.75584312]
average of bagging: 0.8047904103494231
********************************************
```

Random Forest Classifier

```
kf = KFold(n_splits=3)

random_forest = RandomForestClassifier(n_estimators=250, max_depth=7, n_jobs=-1)
rf_results = cross_val_score(random_forest,cancer_data,cancer_target,cv=kf,n_jobs=-1)
print("random forest results:",rf_results)
print("average of rf:",np.mean(rf_results))
print("********************************************")

bagging_rf = BaggingClassifier(random_forest,n_estimators=250,max_samples=0.8,bootstrap=True,n_jobs=-1)
bagging_rf_result = cross_val_score(bagging_rf,cancer_data,cancer_target,cv=kf,n_jobs=-1)
print(" bagging random forest results:",bagging_rf_result)
print("average of bagging rf:",np.mean(bagging_rf_result))
print("********************************************")
```

```
random forest results: [0.92631579 0.97368421 0.95767196]
average of rf: 0.9525573192239859
********************************************
 bagging random forest results: [0.91052632 0.97368421 0.96296296]
average of bagging rf: 0.9490578297595841
********************************************
```

Extra Trees

```
kf = KFold(n_splits=3)
extra_tree =ExtraTreesClassifier(n_estimators=250,max_depth=7, bootstrap=True,n_jobs=-1)
extra_tree_result = cross_val_score(extra_tree,cancer_data,cancer_target,cv=kf,n_jobs=-1)
print(" extra tree results:",extra_tree_result)
print("average of extra tree:",np.mean(extra_tree_result))
```

```
 extra tree results: [0.90526316 0.98421053 0.96296296]
average of extra tree: 0.9508122157244964
```