La segunda evidencia de el avance del proyecto de ensamble de estimadores de clasificadores supervisados

# Evidencia 2

BRAULIO EUGENIO CAMACHO LOYA

Como primer paso, tras la limpieza de los dataset, y generar una estructura que permita acceder a cualquiera, en cualquier momento:

```
#todos los datasets, con la direccion de la clase
def Datasets():
    datasets =[]
    datasets.append(datas("Datasets/bupa.data",6))
    datasets.append(datas("Datasets/diabetic_retinopathy_debrecen.txt", 0))
    datasets.append(datas("Datasets/fertility_Diagnosis2.txt", 9))
    datasets.append(datas("Datasets/haberman.data", 3))
    datasets.append(datas("Datasets/heart.txt", 13))
    datasets.append(datas("Datasets/LSTV", 309))
    datasets.append(datas("Datasets/TrorаricSugery.txt",16))
    datasets.append(datas("Datasets/vertebral_column.txt",6))
    datasets.append(datas("Datasets/train_data.txt",28))
    datasets.append(datas("Datasets/wdbc.txt",1))
    datasets.append(datas("Datasets/wdbcOriginal.txt",1))
    datasets.append(datas("Datasets/wdbcPredictions.txt",1))
    datasets.append(datas("Datasets/chronic_kidney_disease.txt",24))
    datasets.append(datas("Datasets/parkinsons.data",0))
    return datasets
```

Como podemos obesrvar, genera una estructura de datos tipo lista, en un método público de Python, para acceder a ella desde cualquier archivo,junto con la ubicación de la clase, para realizar las pruebas pertinentes.

El código necesario para generar ensambles de clasificadores supervisados consta de 3 partes,

1. Generar la lista de los clasificadores (homogéneos, heterogéneos, o secuenciales)
2. Entrenar los clasificadores en base a un dataset, remuestreando los datos, para generar múltiples subconjuntos de datos, diferentes entre sí, para que exista aleatoriedad y se reduzca la probabilidad de sobreajuste.
3. Probar el resultado, usando, ya sea un conjunto de datos, separados con anterioridad, o con la totalidad de los datos

El conjunto de datos que se separe para probar el modelo resultado, por lo regular es seleccionado por diferentes métodos de validación, como “K-folds”, “Bootstrap validation”, etc.

Dicha estructura se puede observar a continuación:

```
#Carga todos los estimadores, en una lista
estimadores = [
    ('dt',DecisionTreeClassifier(max_depth=5)),
    ('svm', SVC(gamma=1.0, C=1.0, probability=True)),
    ('gp', GaussianProcessClassifier(RBF(1.0))),
    ('3nn',KNeighborsClassifier(n_neighbors=3)),
    ('gnb', GaussianNB())
]
```

```python
#se instancia la clase, y se introdicen los estimadores
class Heterogeneo(BaseEstimator, ClassifierMixin):
    def __init__(self, estimadores):
        self.estimadores = estimadores
```

```python
#se entrenan los estimadores
    def fit(self,X,y):
        X, y = check_X_y(X, y)
        self.classes_ = unique_labels(y)
        for modelo, estimador in self.estimadores:
            estimador.fit(X,y)
        return self
```

```python
#la siguiente estructura sirve para generar predicciones en base a el modelo gwenerado
    def predict(self,X):
        check_is_fitted(self)
        X = check_array(X)
        n_estimadores = len(self.estimadores)
        n_samples = len(X)
        y = np.zeros((n_samples, n_estimadores))
        for i,(modelo,estimador) in enumerate(self.estimadores):
            y[:, i] = estimador.predict(X)
        y = mode(y, axis=1)
        y = [i[0] for i in y[0]]
        return y
```