Rapport

Projet partiel

Équipe 11
Esma Mouine
111 105 194

Définition:

- **Matrice de confusion** est un outil servant à mesurer la qualité d'un système de classification. Chaque colonne représente le nombre d'occurrences d'une classe estimée et chaque ligne représente le nombre d'occurrences d'une classe réelle.

- **La précision** est la proportion de prédictions correctes parmi les points que l'on a prédits positifs. c'est le nombre de prédiction correcte de la classe divisé par le nombre totale de prédiction de la classe.

- **Le rappel** est la proportion de prédiction que l'on a correctement identifiés. C'est le nombre de prédiction correcte de la classe divisée par le nombre de modèle appartenant à la classe

1. Classification avec les K plus proches voisins:

Choix de métrique de distance choisie:

- Le calcul de la distance utilisé est la distance euclidienne, car les données qu'on utilise sont sous forment de vecteurs de tailles différentes. De plus les unités de valeurs des vecteurs sont les mêmes.
- Pseudo code:

```
function train(train, train_labels):
	# initialise les données pour l'entraînement qui vont être utilisé par la fonction predict
	global train_data = train
	global train_data_labels = trains_labels


function Predict(exemple):
	neighbors_indexes = getKNearestNeighbor(exemple, k, train_data) # retourne les k plus proches voisins en fonction de la distance eucledienne
	return getPrediction(neighbors_indexes, train_data_labels)


function getPrediction(neighbors_indexes, train_data_labels):
	classes = list(set(train_data_labels))
	for n in neighbors_indexes:
		for i in range(len(labels)):
			if neighbors_labels[n] == classes[i]:
				votes[i] += 1 # incrémente le vote de 1 à chaque occurrence
	return labels[votes.index(max(votes))] # retourne la classe qui a eu le plus de votes


function test(test, test_labels):
```

```
	#appelle la fonction predict sur tous les éléments du tableau test et affiche la
matrice de confusion, la précision, le rappelle et l'exactitude
```

2. Classification Naïve Bayésienne:

- Pseudo Code:

```
function train(train, train_labels):
	# initialise les données pour l'entraînement qui vont être utilisés, calculer
différentes probabilités de chaque classe en fonction des donnée de train et train_labels
	global classes = list(set(train_labels))
	n, m  = np.shape(train)

	global probabilities = get_prior_probabilies(train_labels) # retourne les
probabilité a priori de chaque classe
	global attributes_probabilities = {}
	# trouver l'occurrence de chaque attribut en fonction de la classe
	for cls in classes:
		indexes = []
		for i in range(n):
			if train_labels[i] == cls:
				indexes.append(i)
		subset = train[indexes, :]
		r, c = np.shape(subset)
		for i in range(c):
			attributes_probabilities[cls][i] += list(subset[:,i])

	# Trouver la probabilité de chaque attribut dans chaque classe classe
	for cls in self.classes:
		for i in range(m):
			attributes_probabilities[cls][i] = occurrences(
								attributes_probabilities[cls][i])


function Predict(exemple):
	results = {}
	for cls in self.classes:
		class_probability = probabilities[cls]
		for i in range(len(exemple)):
			relative_feature_values = attributes_probabilities[cls][i]
			if exemple[i] in relative_feature_values.keys():
				class_probability *= relative_feature_values[exemple[i]]
			else:
				class_probability *= 0
				results[cls] = class_probability

	# retourner la clé de la valeur la plus haute qui représente la prediction
	return max(results, key=lambda i: results[i])


function test(test, test_labels):
	#appelle la fonction predict sur tous les éléments du tableau test et affiche la
matrice de confusion, la précision, le rappelle et l'exactitude
```

3. Discussion des résultat obtenu:

Pour le dataset Iris on retrouve une exactitude beaucoup plus élevé quand on utilise le classificateur des k plus proches voisins (aux alentours de 90 avec le knn et 70 avec le bayésien), alors que pour le dataset des bases d'enregistrement de votes au Congrès, l'exactitude est plus élevée avec le classificateur Bayésien (aux alentours de 70 avec le knn et 90 avec le bayésien). Pour le dataset des

Monks, les résultats sont moins bons que ceux des deux autres est se rapprochent même s'il est plus élevé avec le Knn (les deux sont entre 70 et 80)

4. Comparaison:

Classification avec les K plus proches voisins	Classification Naïve Bayésienne
- Ne fait aucune supposition au sujet de la distribution des données (Pas de probabilité) - Il faut faire une validation pour déterminer la meilleure valeur de k - Approche simple et efficace. Complexité = O(nombre d'instance * nombre d'attribut) - Fonctionne bien sur des problèmes de base - Peut-être lent pour des prédictions à temps réels et s'il y a un grand nombre d'exemples - A besoin de beaucoup de données d'entraînement pour de bons résultats - Utilisation d'une mesure de similarité	- Suppose que les données sont indépendantes - Plus rapide ce n'est que des calculs de probabilité - A besoin de moins de données d'entraînement - Peut être utilisé pour n'importe quels types de données (pas de calcul de distance)

5. Conclusion

Pour conclure, on distingue plusieurs avantages et inconvénients a utilisé les classificateurs bayésiens et le classificateur utilisant l'algorithme des k plus proches voisins. Ce projet a permis de mieux les évaluer et de comprendre le fonctionnement de chacun.