Alumno: Carlos Andrade

Docente: Ing. Diego Quisi.

Materia: SE

Ciclo: 9no

Fecha: 15/05/2020

KNN

K-Nearest-Neighbor es un algoritmo basado en instancia de tipo supervisado de Machine Learning. Puede usarse para clasificar nuevas muestras (valores discretos) o para predecir (regresión, valores continuos). Al ser un método sencillo, es ideal para introducirse en el mundo del Aprendizaje Automático.Implementará un sistema CBR básico para determinar la calidad del vino rojo.

Para detectar la similitudi usamos la formula de Jaccard Similarity

El desarrollo se lo realizara en Python.

def similitudJaccard(valores_vino):

]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

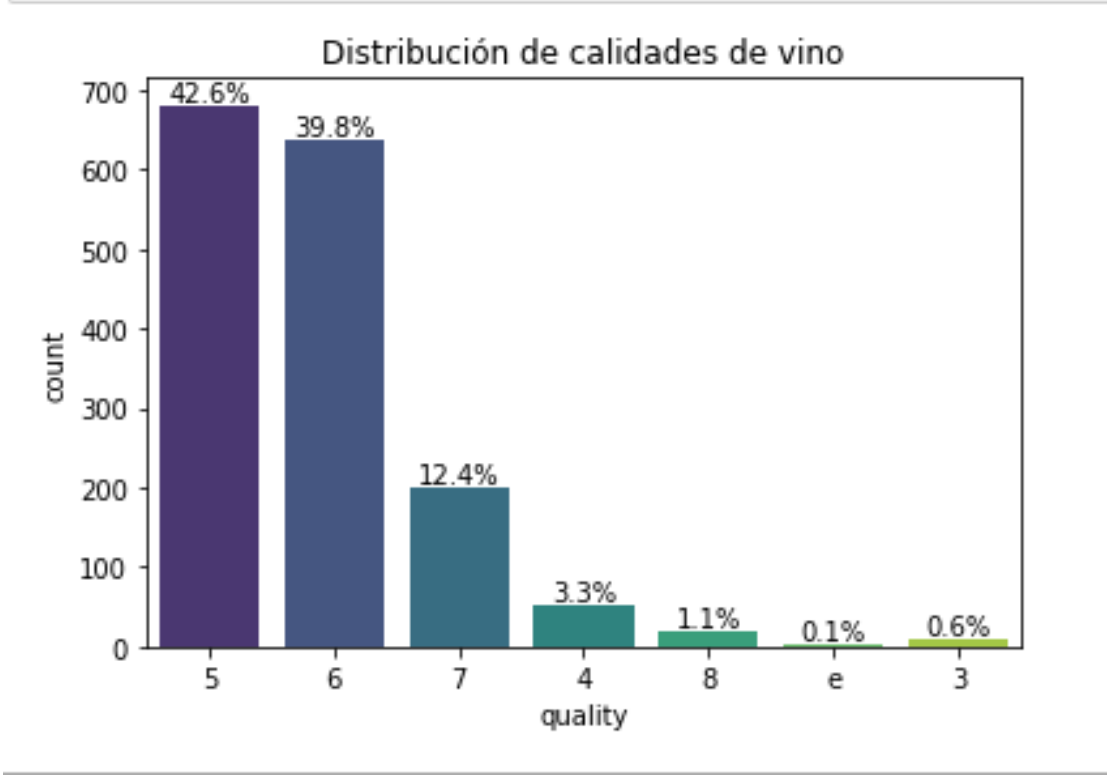
1599 rows × 12 columns

```
def similitud(calidad, set_valores_vino):
    return {
        'calidad': calidad,
        'similitud': len(set_valores_vino.intersection(valores_vino)) /
len(set_valores_vino.union(valores_vino))
    }


list = []
with open("winequality-red.csv", "r") as f:
    valores_vino_data = f.readline()
    while valores_vino_data:
        valores_vino_data = f.readline()
        propiedades_vino = valores_vino_data.split(";")
        calidad = propiedades_vino[-1:][0].replace("\n", "")
        list.append(similitud(calidad, set(map(float, propiedades_vino[:-1]))))
```

```
    return sorted(list, key=lambda item: item['similitud'], reverse=True)
```

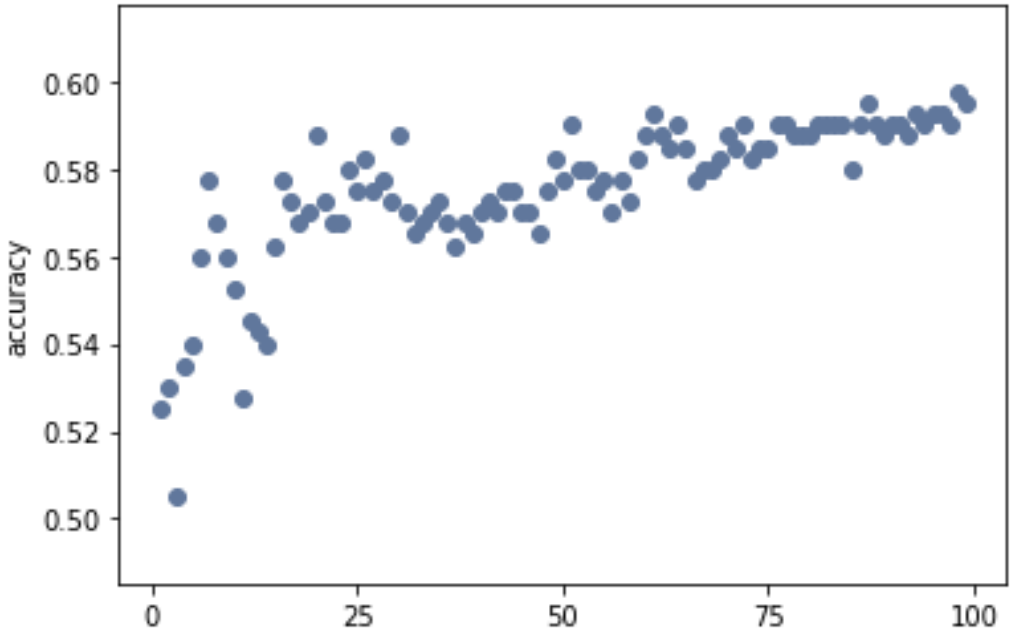
distribución de las calidades de los vinos.

```
plt.title('Distribución de calidades de vino')


for p in ax.patches:
    x=p.get_bbox().get_points()[:,0]
    y=p.get_bbox().get_points()[1,1]
    ax.annotate('{:.1f}%'.format(100.*y/len(df)), (x.mean(), y),
            ha='center', va='bottom') # set the alignment of the text
```




```
k_range = range(1, 100)
scores = []
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors = k)
```

```
    knn.fit(X_train, y_train)
    scores.append(knn.score(X_test, y_test))
plt.figure()
plt.xlabel('k')
plt.ylabel('accuracy')
plt.scatter(k_range, scores)
plt.xticks([0,25,50,75,100])
```



interfaz usuario

```
@app.route('/calcular', methods=['POST'])
def calcular_calidad():
    valores_vino = list(map(float, request.form.values()))
    return render_template("index.html", valores=valores_vino,
lista=similitudJaccard(valores_vino))



@app.route('/')
def index():
    return render_template("index.html")


ax = sns.countplot(df['quality'],palette="viridis")
```

Vino Calidad	Vino Similitud
6	0.2857142857142857
5	0.1875
5	0.1875
6	0.1875
6	0.1875
5	0.1875
5	0.1875
6	0.1875
6	0.1875
6	0.1875
5	0.1875
5	0.1875
5	0.1875
5	0.125
5	0.125
6	0.125

Conclusión:

Muy útil para realizar proyecciones de posibles resultados ya que el algoritmo toma la mayoría de las clases de vecinos y los puede ir puntuando de acuerdo con si similitud pudiendo usar ya sea método Jaccard Similarity tambiem podríamos usar Cosine Similarity y/o Euclidean.