TP2 Laboratorio de Datos

Alamo Malena, Laria Guaza Jeremias, Tag Lucio

November 13, 2023

Contents

1 Introduccion

En el presente trabajo se realizan distintos tipos de entrenamientos, prueba y visualizaciones para crear un modelo que a partir de una imagen prediga que tipo de prenda es.

El entrenamiento se realizo utilizando el dataset "FASHION-MNIST", que posee imagenes pixeladas de distintos tipos de prenda. La informacion de las imagenes se encuentra en una matriz de 28x28 donde cada uno de los valores esta contenido entre 0 y 110 y se corresponde a la intensidad del color negro en el pixel correspondiente (donde 0 es el mas claro y 110 el mas oscuro).

Figure 1: Ejemplos de imagen

2 Analísis Exploratorio

Analizando el dataset llegamos a la conclusión de que los atributos 'pixel' que poseen mayor desvio estandar son más utiles para realizar predicciones que aquellos que tienen menor desvio. Los que tienen mayor desvio pueden ser mas determinantes para decidir a que prenda corresponde una imagen.

En cuanto al parecido de las prendas, para analizar si las remeras son parecidas a los pantalones y pullovers en los datos calculamos la correlacion.

```
Correlación entre remera y pantalón: -0.004204362458998397
Correlación entre remera y pullover: 0.008354487441899148
Correlación entre remera y bota: -0.025806825202094503
```

Figure 2: Correlación entre distintos tipos de prenda

Se observa una mayor correlacion entre remeras y pulloveres que entre remeras y pantalones o remeras y botas. Viendo las imagenes esta diferencia es notoria.

Podemos analizar tambien los tipos de vestidos del dataset. Se observa que son todos vestidos largos y algunos tienen las mismas formas (tienen mangas o cinturas marcadas), pero se pueden encontrar diferencias.

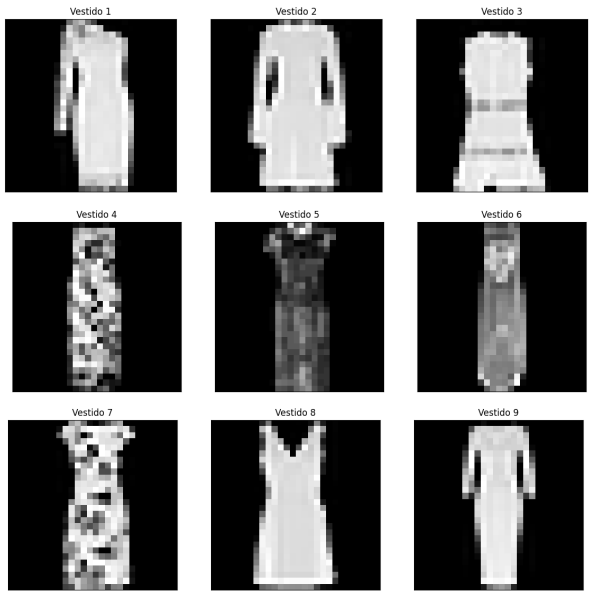

Figure 3: Algunos ejemplos de vestidos

Como el dataset está compuesto por valores de píxeles que representan imágenes, los datos no pueden ser interpretados simplemente leyéndolos, o evaluando métricas como el mínimo, máximo, etc. Para ver diferencias hace falta visualizarlos por nosotros mismos, lo que implica un límite de análisis ya que no podemos visualizar manualmente 60.000 filas en un tiempo considerable. Además el análisis es subjetivo a la persona que lo mire (por ejemplo, pueden haber distintas opiniones acerca de si es más fácil diferenciar una remera de un pullover a una remera de una camisa) y no nos alcanza con reducir el análisis solo a los datos.

3 Clasificacion Binaria

Realizamos un modelo de clasificacion binaria para predecir si una imagen es un pantalon o una remera.

Se tienen 6000 muestras de cada prenda, lo que significa que el dataset está balanceado.

```
# Vemos que esta balanceado
df_pant_rem['label'].value_counts()
✓ 0.0s

0    6000
1    6000
Name: label, dtype: int64
```

Figure 4: Se puede observar 6000 remeras (label=0) y 6000 pantalones (label=1)

Como primera instancia separamos el dataset en train y test y ajustamos un modelo de KNN utilizando una cantidad baja de atributos (3 y 5 pixeles). Elegimos los atributos al azar y analizamos luego la eficacia en el test.

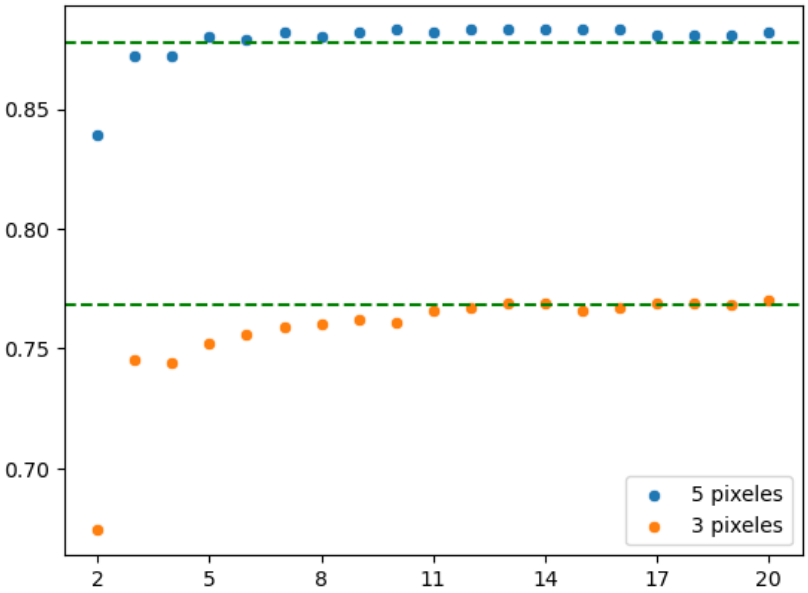

Figure 5: Eficacia en funcion de la cantidad de vecinos utilizando 3 y 5 atributos

Modelamos utilizando un radio de 10 vecinos, ya que en el grafico es el valor a partir del cual la eficacia no se modifica.

Entrenamos un modelo usando todo el conjunto train con 10 vecinos mas cercanos (se puede observar en la ultima figura que es el K mas optimo) y evaluamos en el conjunto de test. Se predijeron de manera correcta 2370 prendas de 2400.
Para un analisis un poco mas extenso, realizamos varios modelos variando cantidad de vecinos y cantidad de atributos. Es importante mencionar que este procedimiento fue realizado determinando una semilla en la cantidad de atributos que se elegian al azar. Observamos los resultados:

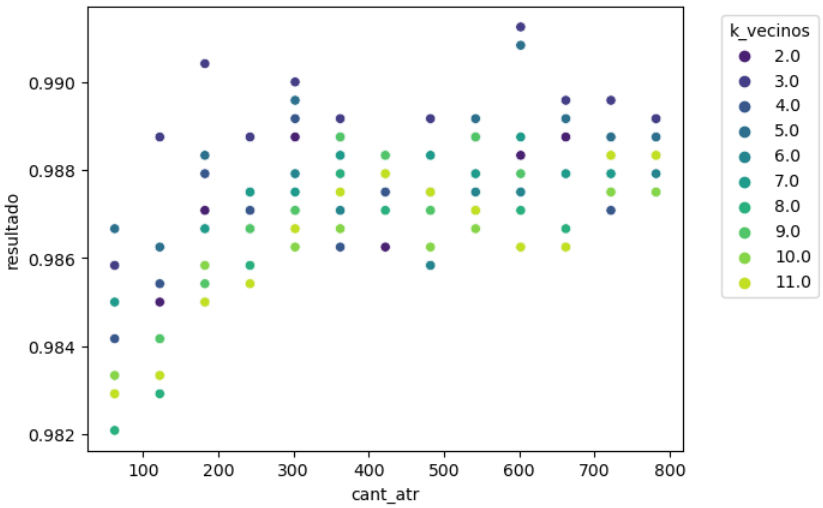

Figure 6: Ejemplos de imagen

4 Multiclase

Para generar un modelo que prediga para cada imagen cual de las 10 prendas es, generamos un modelo de Arbol de Decision.
Para poder decidir la altura y criterio de nuestro arbol, entrenamos distintos modelos variando la altura. Observamos los resultados:

```
alturas = [6,10,14,18,22]
resultados = []
for depth in alturas:
    clf = DecisionTreeClassifier(max_depth=depth, criterion='entropy')
    clf.fit(X_train, y_train)

    y_pred = clf.predict(X_test)
    resultados.append(accuracy_score(y_pred, y_test))
```

```
resultados
```

```
[0.7625, 0.8185, 0.8176666666666667, 0.8116666666666666, 0.8135]
```

Figure 7: Distintos accuracy para distintas alturas con criterio entropia

```
alturas = [6,10,14,18,22]
resultados = []
for depth in alturas:
    clf = DecisionTreeClassifier(max_depth=depth, criterion='gini')
    clf.fit(X_train, y_train)

    y_pred = clf.predict(X_test)
    resultados.append(accuracy_score(y_pred, y_test))
```

```
resultados
```

```
[0.7385,
 0.8129166666666666,
 0.8184166666666667,
 0.8093333333333333,
 0.8019166666666667]
```

Figure 8: Distintos accuracy para distintas alturas con criterio gini

Observamos que los mejores parametros para nuestros modelos son: altura 10 y criterio entropia, altura 14 y criterio entropia, altura 14 y criterio gini.

Para cada uno de ellos entrenamos y testeamos utilizando validacion cruzada con kfolding. Dividimos en 5 pliegues. Observamos los resultados
- Promedio con 5 folds y modelo de altura 10 con entropia como criterio: 0.806625
- Promedio con 5 folds y modelo de altura 14 con gini como criterio: 0.8064375
- Promedio con 5 folds y modelo de altura 14 con entropia como criterio: 0.8055625

Finalmente, observamos que los resultados de los tres modelos son similares. Testeamos utilizando el conjunto de test:
- Accuracy con gini y altura 14: 0.8188333333333333
- Accuracy con entropia y altura 14: 0.81875
- Accuracy con entropia y altura 10: 0.8188333333333333

5 Conclusiones

En el trabajo se pudo observar la dificultad de trabajar con datos que tienen calidad visual y no tanta calidad analitica.
A partir de los modelos entrenados, se pudo observar como se puede llegar a predicciones eficaces utilizando pocos parametros (al contrario de la intuicion). Ademas esto acelera el proceso de entrenamiento de los modelos, ya que se pudo ver que utilizando todos los pixeles como atributos el proceso de entrenamiento se ralentizaba. Esta diferencia de tiempo no era proporcional a la diferencia de resultados.