Reporte prueba Ingeniero de Datos

Nombre: Hugo Pacheco V.

Descripción del problema.

Pipe es especialista en sándwiches, y en sus años de experiencia, elaboró una base de datos de sándwiches de la ciudad de Santiago de Chile, y el resto del mundo (ver @365Sanguchez). Este experto necesita encontrar una receta de sándwich para su futuro restaurant, a partir de la data que el mismo ha registrado.
El objetivo principal de este ejercicio es crear un modelo computacional que permita definir una receta de sándwich que asegure una buena calificación. La base de datos incluye una evaluación a cada sándwich hecha por el propio experto.
Como resultado de la prueba, envíe un reporte describiendo las etapas de su proceso, y los resultados obtenidos, además del código empleado. Debe explicar cómo limpió los datos, como se eligieron y generaron las variables, y como construyó el modelo.

Solución.

Para simplificar se abordó este problema solo considerando las recetas de sándwich (columna ingredientes) y la evaluación de éstas (columna nota). Las notas se evaluaron buenas (1) por sobre nota 2 y malas (0) por menor o igual 2. Transformándolo a un problema binario.

- Primero se cargan y limpias los datos datos desde archivo .csv mediante uso de funciones de biblioteca pandas. Se debe cambiar codificación por error de lectura de archivo, se selecciona las dos columnas de interés y se eliminan de dataframe valores N/A.

```
df = pd.read_csv('sanguchez.csv', sep=';', header=0, encoding='latin-1')#Lee csv con cambio de codificacion
df = df[['Ingredientes', 'nota']]#Filtra columnas de interes
df.dropna(inplace = True)#Elimina valores N/A
```

- Al dataframe resultante se le agrega columna ‘aprobacion’ que asigna 1 para notas mayores 2 y 0 para notas menores o iguales a 2.

```
df['aprobacion'] = [1 if x > 2 else 0 for x in df['nota']]#Agrega columna con evaluacion buena o mala (1 o 0) a partir de las notas
print(df)
```

- Asigna las recetas y las evaluaciones binarias de dataframe a variables individuales para dividir dataset en datos de entrenamiento y datos de prueba. Por medio de función train_test_split de biblioteca sklearn. Deja un 25% de datos para prueba.

```
sentences = df['Ingredientes'].values#Asigna recetas
y = df['aprobacion'].values#Asigna evaluacion de recetas
#Divide dataset, 75% datos entrenamiento y 25% datos test
sentences_train, sentences_test, y_train, y_test = train_test_split(sentences, y, test_size=0.25, random_state = 1000)
print(df.shape, sentences_train.shape, sentences_test.shape, y_train.shape, y_test.shape)
```

- De biblioteca sklearn se utiliza función ‘CountVectorizer’ que creará un índice por cada palabra obtenida de las recetas de sándwich de entrenamiento, no considerando acentos. Con método ‘fit’ crea diccionario a partir de los ingredientes de entrenamiento a número. Se imprime diccionario resultante con 660 palabras.

```
vectorizer = CountVectorizer(strip_accents='ascii')#Asigna funcion vectorizar con preprocesado sin acento y minuscula
#vectorizer = CountVectorizer()
vectorizer.fit(sentences_train)#Crea indices diccionario de palabras a partir de recetas de entrenamiento
print(vectorizer.vocabulary_)#Imprime diccionario
print(len(vectorizer.vocabulary_))
```

- Luego se utiliza este diccionario para transformar las recetas de entrenamiento y prueba a vectores característicos que son un vector numérico de cuantas veces está una palabra en una oración, en este caso una receta de sándwich.

```
X_train = vectorizer.transform(sentences_train)#Crea vectores caracteristicos de recetas de entrenamiento
X_test  = vectorizer.transform(sentences_test)#Crea vectores caracteristicos de recetas de prueba
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
print(type(y_test))
```

- Con las recetas convertidas a vectores característicos se puede ajustar un modelo que pretende imitar el paladar del especialista en sándwich. Se utiliza modelo 'LogisticRegression' de la biblioteca sklearn de buena performance para problemas binarios. Mediante método 'fit' se ajusta el modelo con las recetas de entrenamiento en vectores y las evaluaciones binarias.

```
classifier = LogisticRegression()#Asigna modelo de ajuste binario
classifier.fit(X_train, y_train)#Ajusta modelo binario con vetores caracteristicos y evaluacion de recetas de entrenamiento
```

- Finalmente, con el modelo ajustado se utilizan las recetas de pruebas para tener una predicción de evaluación buena o mala con método 'predict'. Y con método 'score' se compara la predicción del modelo con las evaluaciones reales de los datos de prueba. Obteniendo un 73% de éxito.

```
y_pred = classifier.predict(X_test)#Con modelo ajustado se predice evaluacion de recetas de pruebas
score = classifier.score(X_test, y_test)#Se evalua la predicion de evaluacion del modelo con la evaluacion real de las recetas de p

print("Precisión:", score)#Se imprime indice de aciertos del modelo respecto a la evaluacion real
```

Con este algoritmo tendríamos una buena herramienta para saber si una receta tendría una buena calificación de parte del especialista.

De todas maneras, se puede seguir mejorando el modelo agregando más parámetros como el precio y la georreferenciación de los locales. Y aplicando modelos más complejos.