Nomes: Jonathan Candido da Silva Rodrigues

1) Os dados são carregados para um data frame, após isso foi realizado o agrupamento das features assim contabilizando quantas features temos por tipo de dado (int,float, String), encontramos então que todas as features são float, exceto o 'activity' que é o rótulo que se deseja identificar.

É examinado a divisão das atividades, e como elas são equilibradas não é necessário realizar nenhuma alteração no dataset, assim garantimos que não haverá overfit. Todas as features estão entre -1 e 1, logo não é necessário preocupar que uma feature com valores muito altos influencie a regressão. A coluna da matriz com os rótulos (Activity) é transformada em números inteiros, pois a regressão linear aceita apenas valores numéricos.

2) É calculada matriz de correlação entre os atributos do dataset, mostrando a interdependência entre as variáveis, em seguida a matriz gerada é transformada em uma matriz triangular superior para evitar a duplicidade de correlações.

Após isso é plotado um histograma do valor absoluto da correlação, mostrando que alguns atributos não possuem nenhuma correlação, igual a 0, e poucos atributos possuem correlação total, igual a 1 e são mostrados os 22815 atributos que possuem uma correlação forte, maior que 0.8. Ao analisar as correlações entre as features, podemos afirmar que existem diversas features que poderiam ser desconsideradas no modelo pois apresentam correlação bem próximos a 1.

3) O dataset é divido em treino e teste para ser possível saber a qualidade da predição, 70% para o treino e 30% para o teste.

A divisão do dataset é feita pela classe **StratifiedShuffleSplit** que utliza **n_splits = 1** para definir a quantidade de divisões a serem realizadas, **test_size = 0.3** que varia de 0 a 1 e define a porcentagem do dataset de teste, **random_state** é utilizado para randomizar a divisão dos datasets.

4) A partir do conjunto de treino foi criado um modelo de regressão logística, para isso utilizamos a função **LogisticRegression().fit(X_train, y_train)**, que para o caso de muti classes pode-se utilizar os seguintes algoritmos: **newton-cg, sag, saga** e **lbfgs**. Por isso resolvi treinar e medir a acurácia e tempo para cada algoritmo, alem disso durante os testes percebi que apenas o algoritmo **newton-cg,** converge com o parâmetro padrão da classe, para os outros foi necessário alterar o numero máximo de iterações. para o algoritmo **lbfgs** é necessário um numero de iterações maior que 1000 para que o modelo consiga convergir

que para o caso de multiclasses utiliza o algoritmo de treinamento **multinomial** que é uma forma de regressão logística usada para predizer a classe onde temos mais de 2 classes de classificação. A regressão logística multinomial é uma modificação da regressão logística utilizando a função **softmax** em vez da função **sigmoid**.

5) A acurácia calculada entre os diferentes algoritmos se manteve muito próxima, já o tempo para convergir mudou bastante:
newton-cg: 98,38% - 6.21 s
sag: 98,38% - 14.85 s
saga: 98,41% - 29.44 s
lbfgs: 98,38% - 17.45 s

Entretanto o algoritmo **saga** obteve a melhor acurácia entre eles, mas com a diferença de 0.03%, tendo o maior tempo para convergir 29.44 s, já o algoritmo **newton-cg** foi o mais rápido a convergir demorando apenas 6.21 s e apresentando acurácia de 98.38%. Sendo assim para uma mesma base de dados maior talvez a melhor escolha seja o **newton-cg** pela sua eficácia em convergir e acurácia apresentada.