*#PACKAGES*

**from** **pyspark** **import** SparkContext

sc = SparkContext(master='local[2]')

*#for spark UI*

sc

*#loading packages*

**from** **pyspark.sql** **import** SparkSession

*#spark appname in UI*

spark = SparkSession.builder.appName("Disney's ML code").getOrCreate()

*#loading the dataset*

df = spark.read.csv("Desktop/survey.csv",header=True,inferSchema=True)

*#shows the dataset*

df.show()

*#checking for coloumns*

**print**(df.columns)

df = df.select('Age', 'Gender', 'Country', 'state', 'self\_employed', 'family\_history', 'treatment', 'work\_interfere', 'no\_employees', 'remote\_work', 'tech\_company')

df.show(5)

*# Checking for datatypes*

*# InferSchema=True i.e before*

df.dtypes

*# InferSchema i.e after*

df.dtypes

*# Checking for Schema*

df.printSchema()

*# summary*

**print**(df.describe().show())

*# Value Count*

df.groupBy('Country').count().show()

df.show(5)

**import** **pyspark.ml**

dir(pyspark.ml)

*#loading ml packages*

**from** **pyspark.ml.feature** **import** VectorAssembler,StringIndexer

df.show(4)

*#Unique Values for employment*

df.select('self\_employed').distinct().show()

*# Define the list of categorical columns to be encoded*

cat\_cols = ['Timestamp','Gender', 'Country', 'state', 'self\_employed','family\_history', 'treatment', 'work\_interfere', 'no\_employees', 'remote\_work', 'tech\_company', 'benefits', 'care\_options', 'wellness\_program', 'seek\_help', 'anonymity', 'leave',

'mental\_health\_consequence', 'phys\_health\_consequence', 'coworkers', 'supervisor', 'mental\_health\_interview', 'phys\_health\_interview', 'mental\_vs\_physical', 'obs\_consequence', 'comments']

*# Create a list of StringIndexer stages, one for each categorical column*

indexers = [StringIndexer(inputCol=col, outputCol=col+'\_idx') **for** col **in** cat\_cols]

*# Fit the indexers to the dataframe*

cat\_Encoder = [indexer.fit(df) **for** indexer **in** indexers]

*# Apply the indexers to the dataframe*

encoded\_df = df

**for** encoder **in** cat\_Encoder:

encoded\_df = encoder.transform(encoded\_df)

encoded\_df.show(5)

**print**(encoded\_df.columns)

encoded\_df.dtypes

df2 = encoded\_df.select('Age','Gender\_idx', 'Country\_idx', 'state\_idx', 'self\_employed\_idx', 'family\_history\_idx', 'treatment\_idx',

'work\_interfere\_idx', 'no\_employees\_idx', 'remote\_work\_idx', 'tech\_company\_idx', 'benefits\_idx', 'care\_options\_idx', 'wellness\_program\_idx', 'seek\_help\_idx','anonymity\_idx', 'leave\_idx', 'mental\_health\_consequence\_idx', 'phys\_health\_consequence\_idx', 'coworkers\_idx', 'supervisor\_idx','mental\_health\_interview\_idx', 'phys\_health\_interview\_idx', 'mental\_vs\_physical\_idx',

'obs\_consequence\_idx', 'comments\_idx')

df2.printSchema()

required\_features = ['Age','Gender\_idx', 'Country\_idx', 'state\_idx', 'self\_employed\_idx', 'family\_history\_idx', 'treatment\_idx', 'work\_interfere\_idx', 'no\_employees\_idx', 'remote\_work\_idx', 'tech\_company\_idx', 'benefits\_idx', 'care\_options\_idx', 'wellness\_program\_idx', 'seek\_help\_idx','anonymity\_idx', 'leave\_idx', 'mental\_health\_consequence\_idx', 'phys\_health\_consequence\_idx', 'coworkers\_idx', 'supervisor\_idx', 'mental\_health\_interview\_idx', 'phys\_health\_interview\_idx', 'mental\_vs\_physical\_idx', 'obs\_consequence\_idx', 'comments\_idx']

df2 = df2.toPandas().replace('NA',0).astype(float)

type(df2)

type(df)

*#converting to pyspark dataframe*

new\_df = spark.createDataFrame(df2)

new\_df.show()

*# Checking For DataTypes & Schema*

new\_df.printSchema()

*#VectorAssembler*

vec\_assembler =VectorAssembler(inputCols=required\_features,outputCol='features')

vec\_df = vec\_assembler.transform(new\_df)

vec\_df.show(5)

train\_df,test\_df = vec\_df.randomSplit([0.7,0.3])

train\_df.count()

train\_df.show(4)

**from** **pyspark.ml.classification** **import** LogisticRegression,DecisionTreeClassifier

*# Logistic regression Model*

lr = LogisticRegression(featuresCol='features',labelCol='Country\_idx')

lr\_model = lr.fit(train\_df)

y\_pred = lr\_model.transform(test\_df)

y\_pred.show()

**print**(y\_pred.columns)

y\_pred.select('Country\_idx','rawPrediction', 'probability', 'prediction').show()

**from** **pyspark.ml.evaluation** **import** MulticlassClassificationEvaluator

*#Checking For Accuracy*

multi\_evaluator = MulticlassClassificationEvaluator(labelCol='Country\_idx',metricName='accuracy')

multi\_evaluator.evaluate(y\_pred)

**from** **pyspark.mllib.evaluation** **import** MulticlassMetrics

lr\_metric = MulticlassMetrics(y\_pred['Country\_idx', 'prediction'].rdd)

dir(lr\_metric)

**print**("Accuracy",lr\_metric.accuracy)

**print**("Precision",lr\_metric.precision(1.0))

**print**("Recall",lr\_metric.recall(1.0))

**print**("F1Score",lr\_metric.fMeasure(1.0))

*# Saving Model*

lr\_model.save("Disneylr\_model\_30")

lr\_model.write().save("Disneylr\_model")