Universitatea din București
Facultatea de Matematică și Informatică

Proiect Big Data

Analiza setului de date

Wine Quality

oferit de

University of California Irvine

Student:
Dobre Mihaela-Beatrice, grupa 405

Cuprins

1. Introducere

a) Prezentarea setului de date

Setul de date folosit în cadrul acestui proiect se află pe site-ul Universității din California Irvine la secțiunea Machine Learning Repository. Acesta conține informații despre multiple caracteristici care influențează calitatea vinului din nordul Portugaliei, reprezentând o atât o problemă de clasificare, cât și o problemă de regresie. Avem date atât pentru calitatea vinului roșu, cât și pentru cel alb. Am ales sa folosesc setul de date pentru calitatea vinului alb deoarece conține considerabil mai multe date.

Atributele regăsite în setul de date sunt următoarele:

1. fixed acidity - aciditatea fixă
2. volatile acidity - aciditatea volatilă
3. citric acid - acidul citric
4. residual sugar - zahărul rezidual
5. chlorides - cloruri
6. free sulfur dioxide - dioxid de sulf liber
7. total sulfur dioxide - dioxidul total de sulf
8. density - densitatea
9. pH
10. sulphates - sulfați
11. alcohol - alcool

Variabila de output (bazată pe toate celelalte):

12. quality (scor între 0 și 10) - calitatea vinului

Setul de date și informațiile despre acesta pot fi regăsite la următorul link:
https://archive.ics.uci.edu/ml/datasets/Wine+Quality

b) Enunțarea obiectivelor

În cadrul acestui proiect, se propune înțelegerea și analiza setului de date, observarea influenței fiecărei caracteristici asupra calității vinului, cât și curățarea acestuia pentru a putea fi folosit în crearea modelor de Machine Learning și de Deep Learning. Se propune folosirea tehnicilor pentru înțelegerea conceptelor de Big Data, precum Data Pipelines și Spark Streaming.

2. Script pentru procesarea, pregătirea, curățarea și transformarea datelor

Proiectul este realizat în mediul Google Collab, care oferă un environment interactiv pentru rularea de cod Python în Cloud. Librăriile folosite permit accesul la funcționalitățile oferite de Spark SQL și Dataframes, dar și Pandas, matplotlib și alte funcții pentru partea de Machine Learning din pyspark.ml și Tensorflow.

Pentru început, se creează un obiect SparkSession și un Dataframe prin citirea setului de date CSV:

```
spark = SparkSession.builder.appName("Project_AirQuality").getOrCreate()
df =
spark.createDataFrame(pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv", delimiter=';'))
```

Apoi au fost rulate mai multe comenzi pentru a explora setul de date, precum:

```
df.count()
df.show()
df.describe().show()
df.printSchema()
```

```
[ ]  df.printSchema()

root
 |-- fixed acidity: double (nullable = true)
 |-- volatile acidity: double (nullable = true)
 |-- citric acid: double (nullable = true)
 |-- residual sugar: double (nullable = true)
 |-- chlorides: double (nullable = true)
 |-- free sulfur dioxide: double (nullable = true)
 |-- total sulfur dioxide: double (nullable = true)
 |-- density: double (nullable = true)
 |-- pH: double (nullable = true)
 |-- sulphates: double (nullable = true)
 |-- alcohol: double (nullable = true)
 |-- quality: long (nullable = true)
```

Am analizat setul de date prin rularea unor query-uri folosind Dataframes și Spark SQL:

- Afișarea vinurilor cu un *pH* mai mare sau egal cu 3 și cu o concentrație de alcool mai mică de 10, ordonate descrescător în funcție de calitate.

```
df.createOrReplaceTempView("wines")
queryDF = spark.sql("SELECT * FROM wines WHERE ph >= 3 and alcohol < 10 ORDER BY quality DESC")
queryDF.show()
```

fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
7.0	0.12	0.29	10.3	0.039	41.0	98.0	0.99564	3.19	0.38	9.8	8
6.7	0.26	0.39	1.1	0.04	45.0	147.0	0.9935	3.32	0.58	9.6	8
8.1	0.17	0.44	14.1	0.053	43.0	145.0	1.0006	3.28	0.75	8.8	8
7.0	0.12	0.29	10.3	0.039	41.0	98.0	0.99564	3.19	0.38	9.8	8
7.6	0.2	0.3	14.2	0.056	53.0	212.5	0.999	3.14	0.46	8.9	8
7.6	0.2	0.3	14.2	0.056	53.0	212.5	0.999	3.14	0.46	8.9	8
8.1	0.17	0.44	14.1	0.053	43.0	145.0	1.0006	3.28	0.75	8.8	8
7.6	0.2	0.3	14.2	0.056	53.0	212.5	0.999	3.14	0.46	8.9	8
7.6	0.2	0.3	14.2	0.056	53.0	212.5	0.999	3.14	0.46	8.9	8
7.8	0.18	0.46	12.6	0.042	41.0	143.0	1.0	3.24	0.76	8.5	8
7.6	0.2	0.3	14.2	0.056	53.0	212.5	0.999	3.14	0.46	8.9	8
6.9	0.21	0.28	2.4	0.056	49.0	159.0	0.9944	3.02	0.47	8.8	8
7.6	0.2	0.3	14.2	0.056	53.0	212.5	0.999	3.14	0.46	8.9	8
7.0	0.12	0.29	10.3	0.039	41.0	98.0	0.99564	3.19	0.38	9.8	8
7.0	0.3	0.38	14.9	0.032	60.0	181.0	0.9983	3.18	0.61	9.3	7
7.4	0.155	0.34	2.3	0.045	73.5	214.0	0.9934	3.18	0.61	9.9	7
6.5	0.24	0.32	7.6	0.038	48.0	203.0	0.9958	3.45	0.54	9.7	7
7.4	0.155	0.34	2.3	0.045	73.5	214.0	0.9934	3.18	0.61	9.9	7
6.5	0.24	0.32	7.6	0.038	48.0	203.0	0.9958	3.45	0.54	9.7	7
7.0	0.17	0.31	4.8	0.034	34.0	132.0	0.9944	3.36	0.48	9.6	7

only showing top 20 rows

- Afișarea numărului de tipuri de vin corespunzătoare fiecărei calități în ordinea crescătoare a calității.

```
grouped = df.groupBy(df.quality).count().sort("quality")
grouped.show(truncate=False)
```

quality	count
3	20
4	163
5	1457
6	2198
7	880
8	175
9	5

- Afișarea cantității de zahăr rezidual pentru tipurile de vin cu concentrația maximă de alcool.

```
maxAlcohol = spark.sql("SELECT MAX(alcohol) FROM wines").collect()[0][0]
df.select('residual sugar').filter(col('alcohol') == maxAlcohol).collect()
```

```
[Row(residual sugar=1.6)]
```

Pregătirea și curățarea datelor

Pentru pregătirea datelor, au fost necesare mai multe etape:

- Înlocuirea spațiilor din numele caracteristicilor cu _ pentru claritate.

```
cols = df.columns
for c in cols:
    df = df.withColumnRenamed(c, c.replace(' ', '_'))
```

```
['fixed_acidity',
 'volatile_acidity',
 'citric_acid',
 'residual_sugar',
 'chlorides',
 'free_sulfur_dioxide',
 'total_sulfur_dioxide',
 'density',
 'pH',
 'sulphates',
 'alcohol',
 'quality']
```

- După cum s-a putut observa în analiza setului de date, range-ul calităților este de la 3 la 9, așa că vom mapa coloana quality pentru a fi între 0 și 6 pentru a lucra mai ușor în cadrul modelelor.

```
df = df.withColumn('quality', col('quality') - 3)
```

- Testarea nulității caracteristicilor sau a valorilor lipsă. Nu există astfel de valori în setul de date.

```
df.select([count(when(isnan(col(c)) | col(c).isNull(), c)).alias(c) for c in
df.columns]).show()
```

```
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+---+---------+-------+-------+
|fixed_acidity|volatile_acidity|citric_acid|residual_sugar|chlorides|free_sulfur_dioxide|total_sulfur_dioxide|density| pH|sulphates|alcohol|quality|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+---+---------+-------+-------+
|            0|               0|          0|             0|        0|                  0|                   0|      0|  0|        0|      0|      0|
+-------------+----------------+-----------+--------------+---------+-------------------+--------------------+-------+---+---------+-------+-------+
```

- Ștergerea duplicatelor din setul de date. Înainte de acest lucru, setul de date avea 4898 de intrări, iar după ștergere au rămas 3961 intrări.

```
if df.count() > df.dropDuplicates(df.columns).count():
    df = df.dropDuplicates(df.columns)
```

Mai apoi, am afișat anumite informații despre corelația caracteristicilor din setul de date:

- Reprezentarea matricei de corelație

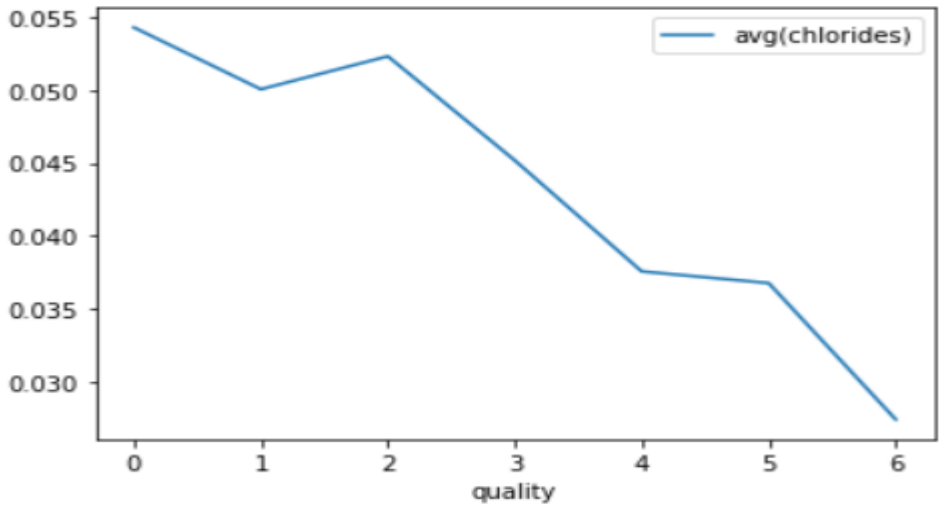
```
df.toPandas().corr().style.background_gradient("BuPu")
```

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alcohol	quality
fixed_acidity	1.000000	-0.019214	0.298959	0.083620	0.024036	-0.058396	0.082425	0.266091	-0.431274	-0.017453	-0.110788	-0.124636
volatile_acidity	-0.019214	1.000000	-0.163228	0.098340	0.086287	-0.102471	0.102315	0.060603	-0.046954	-0.021150	0.046815	-0.190678
citric_acid	0.298959	-0.163228	1.000000	0.106269	0.132590	0.091681	0.122845	0.160076	-0.183015	0.049442	-0.076514	0.007065
residual_sugar	0.083620	0.098340	0.106269	1.000000	0.076091	0.306835	0.409583	0.820498	-0.165997	-0.020503	-0.398167	-0.117339
chlorides	0.024036	0.086287	0.132590	0.076091	1.000000	0.101272	0.191145	0.253088	-0.090573	0.017871	-0.356928	-0.217739
free_sulfur_dioxide	-0.058396	-0.102471	0.091681	0.306835	0.101272	1.000000	0.619437	0.294638	-0.007750	0.037932	-0.251768	0.010507
total_sulfur_dioxide	0.082425	0.102315	0.122845	0.409583	0.191145	0.619437	1.000000	0.536868	0.008239	0.136544	-0.446643	-0.183356
density	0.266091	0.060603	0.160076	0.820498	0.253088	0.294638	0.536868	1.000000	-0.063734	0.082048	-0.760162	-0.337805
pH	-0.431274	-0.046954	-0.183015	-0.165997	-0.090573	-0.007750	0.008239	-0.063734	1.000000	0.142353	0.093095	0.123829
sulphates	-0.017453	-0.021150	0.049442	-0.020503	0.017871	0.037932	0.136544	0.082048	0.142353	1.000000	-0.022850	0.053200
alcohol	-0.110788	0.046815	-0.076514	-0.398167	-0.356928	-0.251768	-0.446643	-0.760162	0.093095	-0.022850	1.000000	0.462869
quality	-0.124636	-0.190678	0.007065	-0.117339	-0.217739	0.010507	-0.183356	-0.337805	0.123829	0.053200	0.462869	1.000000

În cadrul acestei reprezentări, câmpurile închise la culoare reprezintă o corelație mai mare între date. De exemplu, se remarcă o legătură destul de puternică între densitate și zahărul rezidual.

- Reprezentarea legăturii și a influenței clorurilor pentru calitatea vinului.

```
grouped = df.groupBy(df.quality).mean().sort("quality")
grouped.toPandas().plot(x="quality", y=['avg(chlorides)'])
```



Din graficul de mai sus, se poate deduce că în principal, odată cu scăderea nivelului de cloruri în compoziția vinului, crește calitatea acestuia.

Transformarea datelor

VectorAssembler este un transformator ce ajută la combinarea unei liste date de coloane într-un singur vector coloană, ceea ce este util pentru combinarea caracteristicilor necesare antrenării unui model ML. În cazul meu, am îmbinat toate coloanele de caracteristici.

```
cols = df.columns
cols.remove('quality')
va = VectorAssembler(inputCols = cols, outputCol='features')
df = va.transform(df)
```

StandardScaler este folosit pentru a redimensiona distribuția valorilor astfel încât media valorilor observate este 0, iar abaterea standard este 1. Astfel, toate datele se regăsesc în aceeași scară.

```
scaler = StandardScaler(inputCol='features', outputCol='scaledFeatures', withStd=True,
withMean=False)
scaler_model = scaler.fit(df)
df = scaler_model.transform(df)
```

Aceste rezultate vor fi folosite în continuare pentru modelele de ML și de Deep Learning.

3. Spark ML

Determinarea scorului de calitate al vinului alb poate reprezenta atât o problemă de clasificare, cât și de regresie, dar în cadrul acestui proiect am ales să o tratez drept o clasificare. Sunt 7 clase posibile în care pot fi încadrate datele, de la 0 la 6, 6 fiind scorul maxim pentru calitatea vinului în acest caz.

Pentru început, datele sunt împărțite într-un set de antrenare și unul de test cu ajutorul funcției **randomSplit,** în proporție de 70%-30%.

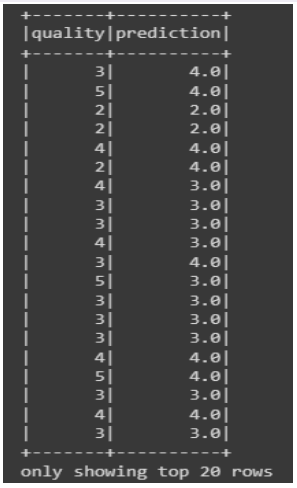
```
train, test = df.select('scaledFeatures', 'quality').randomSplit([.7, .3],
seed=23)
```

3.1. Regresia logistică

În acest caz a fost folosită Regresia Logistică multinomială. Aceasta este utilizată pentru a prezice plasarea categorială sau probabilitatea apartenenței la categorie pe o variabilă dependentă pe baza mai multor variabile independente.

```
lr = LogisticRegression(featuresCol='scaledFeatures',labelCol='quality',
```

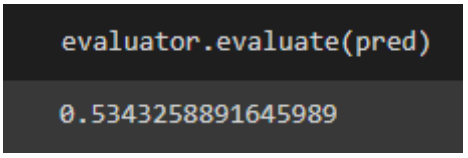
```
family='multinomial')
lr_model=lr.fit(train)
pred = lr_model.evaluate(test).predictions
pred = pred.select('quality', 'prediction')
```



```
+-------+----------+
|quality|prediction|
+-------+----------+
|      3|       4.0|
|      5|       4.0|
|      2|       2.0|
|      2|       2.0|
|      4|       4.0|
|      2|       4.0|
|      4|       3.0|
|      3|       3.0|
|      3|       3.0|
|      4|       3.0|
|      3|       4.0|
|      5|       3.0|
|      3|       3.0|
|      3|       3.0|
|      3|       3.0|
|      4|       4.0|
|      5|       4.0|
|      3|       3.0|
|      4|       4.0|
|      3|       3.0|
+-------+----------+
only showing top 20 rows
```

Evaluarea modelului cu ajutorul **MulticlassClassificationEvaluator**, un evaluator clasic pentru problemele de clasificare cu mai multe clase.

```
evaluator = MulticlassClassificationEvaluator(predictionCol='prediction',
labelCol='quality', metricName='accuracy')
evaluator.evaluate(pred)
```

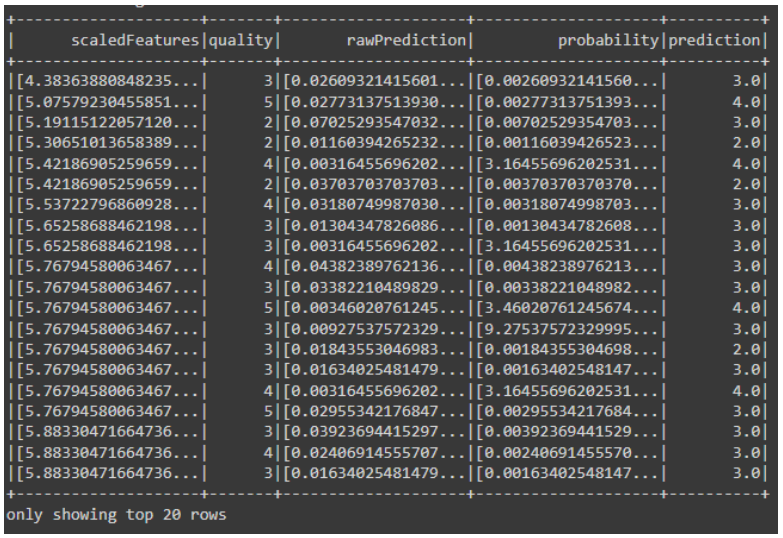


```
evaluator.evaluate(pred)

0.5343258891645989
```

3.2. Random Forest

Algoritmul Random Forest este o metodă de clasificare bazată pe ansambluri ce operează cu noțiunea de arbore de decizie. Metodele bazate pe ansambluri funcționează pe principiul conform căruia un grup de „clasificatori slabi" pot forma, împreună, un clasificator puternic. În cadrul Random Forest, se generează mai mulți arbori de decizie pornind de la submulțimi ale setului de date inițial. Fiecare arbore astfel generat se aplică pe instanța ce trebuie clasificată, rezultând câte o clasă pentru fiecare arbore. Rezultatul final al clasificării este dat de clasa majoritară.

```
rf = RandomForestClassifier(featuresCol='scaledFeatures', labelCol='quality',
predictionCol='prediction', numTrees=10, seed=42)
rf_model = rf.fit(train)
pred = rf_model.evaluate(test).predictions
pred.show()
```



```
+--------------------+-------+--------------------+--------------------+----------+
|      scaledFeatures|quality|       rawPrediction|         probability|prediction|
+--------------------+-------+--------------------+--------------------+----------+
|[4.38363880848235...|      3|[0.02609321415601...|[0.00260932141560...|       3.0|
|[5.07579230455851...|      5|[0.02773137513930...|[0.00277313751393...|       4.0|
|[5.19115122057120...|      2|[0.07025293547032...|[0.00702529354703...|       3.0|
|[5.30651013658389...|      2|[0.01160394265232...|[0.00116039426523...|       2.0|
|[5.42186905259659...|      4|[0.00316455696202...|[3.16455696202531...|       4.0|
|[5.42186905259659...|      2|[0.03703703703703...|[0.00370370370370...|       2.0|
|[5.53722796860928...|      4|[0.03180749987030...|[0.00318074998703...|       3.0|
|[5.65258688462198...|      3|[0.01304347826086...|[0.00130434782608...|       3.0|
|[5.65258688462198...|      3|[0.00316455696202...|[3.16455696202531...|       3.0|
|[5.76794580063467...|      4|[0.04382389762136...|[0.00438238976213...|       3.0|
|[5.76794580063467...|      3|[0.03382210489829...|[0.00338221048982...|       3.0|
|[5.76794580063467...|      5|[0.00346020761245...|[3.46020761245674...|       4.0|
|[5.76794580063467...|      3|[0.00927537572329...|[9.27537572329995...|       3.0|
|[5.76794580063467...|      3|[0.01843553046983...|[0.00184355304698...|       2.0|
|[5.76794580063467...|      3|[0.01634025481479...|[0.00163402548147...|       3.0|
|[5.76794580063467...|      4|[0.00316455696202...|[3.16455696202531...|       4.0|
|[5.76794580063467...|      5|[0.02955342176847...|[0.00295534217684...|       3.0|
|[5.88330471664736...|      3|[0.03923694415297...|[0.00392369441529...|       3.0|
|[5.88330471664736...|      4|[0.02406914555707...|[0.00240691455570...|       3.0|
|[5.88330471664736...|      3|[0.01634025481479...|[0.00163402548147...|       3.0|
+--------------------+-------+--------------------+--------------------+----------+
only showing top 20 rows
```

```
evaluator.evaluate(pred)
```

```
0.5376344086021505
```

3.3. Linear Support Vector Classification

Nu toate metodele de clasificare suportă clasificare multiclasă. Printre aceste metode se numără și SVC(Support Vector Classification). Pentru a putea face acest lucru posibil, se folosește strategia One vs Rest care împarte clasificarea multiclasă într-o problemă de clasificare binară per clasă. Adică, se ia fiecare clasă pe rând și se compară cu celelalte clase care sunt reprezentate ca una singură.

```
svm = LinearSVC(regParam=0.01)
ovr = OneVsRest(featuresCol='scaledFeatures', labelCol='quality', classifier=svm)
ovr_model = ovr.fit(train)
pred = ovr_model.transform(test)
```

```
[ ]  evaluator.evaluate(pred)

     0.5169561621174524
```

4. Data Pipeline

Un Data Pipeline conține transformatori și estimatori, iar acesta permite menținerea fluxului de date prin toate transformările revelante, necesare pentru obținerea rezultatului final. Pentru aceasta, trebuie definite inițial toate etapele pipeline-ului.

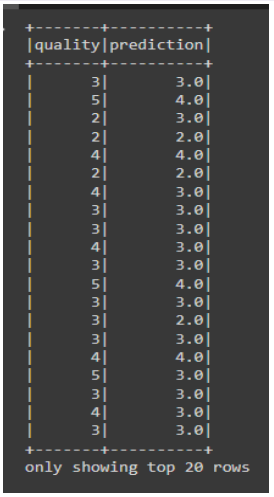
Adăugarea în pipeline a vectorizerului, scalerului și a modelului Random Forest.

```
pipeline = Pipeline(stages=[va, scaler, rf])
```

```
df = df.drop('features', 'scaledFeatures')
# impartirea setului de date in train si test
train, test = df.randomSplit([.7, .3], seed=23)
```

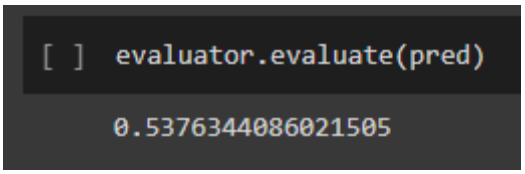
Antrenarea și testarea modelului de pipeline

```
fitted_model = pipeline.fit(train)
pred = fitted_model.transform(test)
pred.select('quality', 'prediction').show()
```



```
+-------+----------+
|quality|prediction|
+-------+----------+
|      3|       3.0|
|      5|       4.0|
|      2|       3.0|
|      2|       2.0|
|      4|       4.0|
|      2|       2.0|
|      4|       3.0|
|      3|       3.0|
|      3|       3.0|
|      4|       3.0|
|      3|       3.0|
|      5|       4.0|
|      3|       3.0|
|      3|       2.0|
|      3|       3.0|
|      4|       4.0|
|      5|       3.0|
|      3|       3.0|
|      4|       3.0|
|      3|       3.0|
+-------+----------+
only showing top 20 rows
```

Evaluarea pipeline-ului:



```
[ ]  evaluator.evaluate(pred)

0.5376344086021505
```

5. Metoda Deep Learning

Împărțirea datelor în date de training și date de test cu ajutorul metodei **train_test_split**, în proporție de 70%-30% ca și până acum, plus stratificare după clasa (quality) deoarece aceasta poate îmbunătăți în final acuratețea per clasa.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y,
random_state=23)
```

Transformarea datelor în date categorice.

```
y_train = to_categorical(y_train['quality'].to_numpy(), num_classes=7)
y_test = to_categorical(y_test['quality'].to_numpy(), num_classes=7)
```

Pentru arhitectura rețelei am folosit un model secvențial în care am înlănțuit 4 layere dense cu activare ReLU, cu număr variabil de neuroni, urmate de un layer de ieșire de 7 neuroni, cu activare softmax pentru a obține probabilitățile celor 7 clase.

Pentru optimizare am folosit Adam deoarece tinde să conveargă cel mai rapid și precis către un rezultat acceptabil, în comparație cu SGD de exemplu. Epocile și mărimea batch-ului au fost alese euristic în urma experimentelor. Am experimentat și cu cateva layere de Dropout pentru regularizare, însă nu îmbunătățeau acuratețea sau loss-ul.

Crearea, antrenarea și compilarea unei rețele neurale adaptate problemei de clasificare.

```
model = Sequential()
model.add(InputLayer(input_shape=(11,)))
model.add(Dense(100, activation='relu'))
model.add(Dense(50, activation='relu'))
model.add(Dense(28, activation='relu'))
model.add(Dense(14, activation='relu'))
# model.add(Dropout(0.3))
model.add(Dense(7, activation="softmax"))

model.summary()

model.compile(optimizer=Adam(0.001), loss='categorical_crossentropy',
metrics=['categorical_accuracy'])

model.fit(X_train, y_train, batch_size=64, epochs=200)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 100)               1200

 dense_1 (Dense)             (None, 50)                5050

 dense_2 (Dense)             (None, 28)                1428

 dense_3 (Dense)             (None, 14)                406

 dense_4 (Dense)             (None, 7)                 105

=================================================================
Total params: 8,189
Trainable params: 8,189
Non-trainable params: 0
```

Prezicerea pe datele de test și afișarea acurateții modelului.

```
preds = model.predict(X_test)
m = CategoricalAccuracy()
m.update_state(y_test, preds)
m.result().numpy()
```

```
0.5121951
```

6. Spark Streaming

Spark Streaming este o extensie a Spark API care permite procesarea stream-urilor de date live. Acesta poate primi input-uri de date din multiple surse pe care le împarte în batch-uri, care sunt apoi procesate de motorul Spark pentru a genera fluxul final de rezultate.

Pentru început, se creează un StreamingContext din Spark Context-ul actual care va citi din stream batch-uri o dată la 3 secunde în cazul nostru. Apoi setam path-ul de unde vor fi preluate date și modalitatea lor de procesare.

```
ssc = StreamingContext(spark.sparkContext, 3)
lines = ssc.textFileStream('/content/')
rows = lines.map(lambda line: [float(x) for x in line.split(",")])
rows.foreachRDD(process_stream)
```

Creăm funcția ce va procesa fiecare fișier ce este transferat în folder-ul *content*, și îl va trece în formă de DataFrame prin pipeline-ul declarat și antrenat anterior cu RandomForest, afișând ulterior rezultatul transformării (predicției):

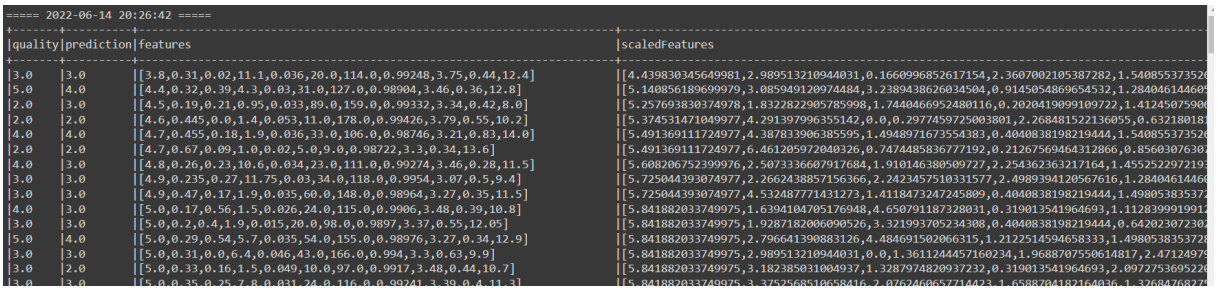
```
def f(x):
    d = {}
    for i in range(len(x)):
        d[columns[i]] = x[i]
    return d

def process_stream(time, record):
    print(f"===== {time} =====")
    try:
        df = record.map(lambda x: Row(**f(x))).toDF()
        pred = fitted_model.transform(df).select(["quality", "prediction", "features",
"scaledFeatures"])
        pred.show(pred.count(), False)
    except Exception as e:
        print(e)
```

Salvăm datele de test în folderul *content* pentru a fi folosite ca input în cadrul streaming-ului de date:

```
test.toPandas().to_csv("/content/wines.csv", index=False, header=False)
```

Pornirea contextului:

```
ssc.start()
```



```
===== 2022-06-14 20:26:42 =====
+-------+----------+--------------------------------------------------------------------+--------------------------------------------------------------------
|quality|prediction|features                                                            |scaledFeatures
+-------+----------+--------------------------------------------------------------------+--------------------------------------------------------------------
|3.0    |3.0       |[3.8,0.31,0.02,11.1,0.036,20.0,114.0,0.99248,3.75,0.44,12.4]        |[4.4398303456499 81,2.989513210944031,0.16609968526171 54,2.360700210538 7282,1.540855373526
|5.0    |4.0       |[4.4,0.32,0.39,4.3,0.03,31.0,127.0,0.98904,3.46,0.36,12.8]          |[5.140856189699979,3.085949120974484,3.2389438626034504,0.9145054869654532,1.284046144605
|2.0    |3.0       |[4.5,0.19,0.21,0.95,0.033,89.0,159.0,0.99332,3.34,0.42,8.0]         |[5.257693830374978,1.8322822905785998,1.7440466952480116,0.2020419099109722,1.41245075900
|2.0    |2.0       |[4.6,0.445,0.0,1.4,0.053,11.0,178.0,0.99426,3.79,0.55,10.2]         |[5.374531471049977,4.291397996355142,0.0,0.2977459725003801,2.268481522136055,0.632180181
|4.0    |4.0       |[4.7,0.455,0.18,1.9,0.036,33.0,106.0,0.98746,3.21,0.83,14.0]        |[5.491369111724977,4.387833906385595,1.4948971673554383,0.4040838198219444,1.540855373526
|2.0    |2.0       |[4.7,0.67,0.09,1.0,0.02,5.0,9.0,0.98722,3.3,0.34,13.6]              |[5.491369111724977,6.461205972040326,0.7474485836777192,0.2126756946431 2866,0.856030763017
|4.0    |3.0       |[4.8,0.26,0.23,10.6,0.034,23.0,111.0,0.99274,3.46,0.28,11.5]        |[5.608206752399976,2.5073336607917684,1.9101463805097 27,2.254362363217164,1.455252297219
|3.0    |3.0       |[4.9,0.235,0.27,11.75,0.03,34.0,118.0,0.9954,3.07,0.5,9.4]          |[5.725044393074977,2.2662438857156366,2.2423457510331577,2.4989394120567616,1.284046144660
|3.0    |3.0       |[4.9,0.47,0.17,1.9,0.035,60.0,148.0,0.98964,3.27,0.35,11.5]         |[5.725044393074977,4.532487771431273,1.4118473247245809,0.4040838198219444,1.498053835372
|4.0    |3.0       |[5.0,0.17,0.56,1.5,0.026,24.0,115.0,0.9906,3.48,0.39,10.8]          |[5.841882033749975,1.6394104705176948,4.650791187328031,0.3190135419646 93,1.112839991991
|3.0    |3.0       |[5.0,0.2,0.4,1.9,0.015,20.0,98.0,0.9897,3.37,0.55,12.05]            |[5.841882033749975,1.9287182006090526,3.321993705234308,0.4040838198219444,0.642023072302
|5.0    |4.0       |[5.0,0.29,0.54,5.7,0.035,54.0,155.0,0.98976,3.27,0.34,12.9]         |[5.841882033749975,2.796641390883126,4.484691502066315,1.2122514594658333,1.498053835372
|3.0    |3.0       |[5.0,0.31,0.0,6.4,0.046,43.0,166.0,0.994,3.3,0.63,9.9]              |[5.841882033749975,2.989513210944031,0.0,1.3611244457160234,1.9688707550614817,2.47124979
|3.0    |2.0       |[5.0,0.33,0.16,1.5,0.049,10.0,97.0,0.9917,3.48,0.44,10.7]           |[5.841882033749975,3.182385031004937,1.3287974820937232,0.3190135419646 93,2.097275369522
|3.0    |3.0       |[5.0,0.35,0.25,7.8,0.031,24.0,116.0,0.99241,3.39,0.4,11.3]          |[5.841882033749975,3.375256851065841 6,2.0762460657714423,1.6588704182164036,1.326847682
```

Așteptarea opririi streaming-ului care se poate face de exemplu prin folosirea comenzii CTRL+C în terminal:

```
ssc.awaitTermination()
```