```
import pyspark
from pyspark.sql import SparkSession
from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import VectorAssembler
import matplotlib.pyplot as plt
import seaborn as sns

from pylab import *
from pyspark.sql.functions import udf, concat, col, lit
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession, SQLContext

from pyspark.sql.types import *
import pyspark.sql.functions as F
sc = SparkContext.getOrCreate(SparkConf().setMaster("local[*]"))
from pyspark.sql import SparkSession
spark = SparkSession \
    .builder \
    .getOrCreate()
sqlContext = SQLContext(sc)
```

```
/databricks/spark/python/pyspark/sql/context.py:117: FutureWarning:
Deprecated in 3.0.0. Use SparkSession.builder.getOrCreate() instead.
  warnings.warn(
```

```
spark = SparkSession.builder.appName("fishes").getOrCreate()
```

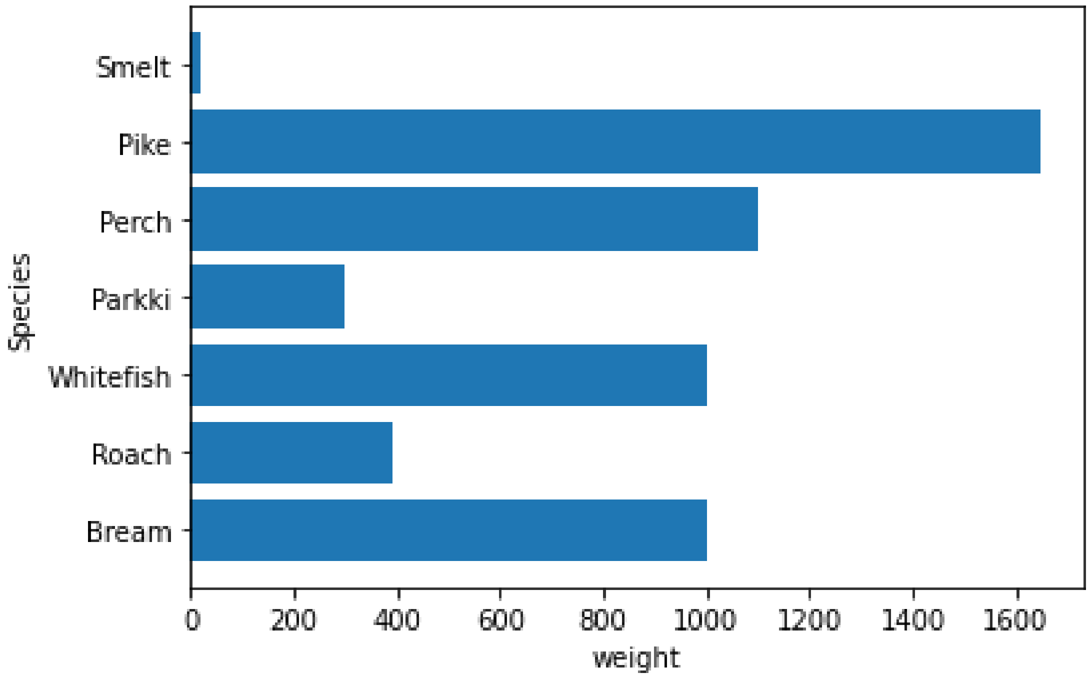
```
dataset = spark.read.csv("/FileStore/tables/Fish-1.csv",
inferSchema=True, header =True)
dataset.printSchema()
```

```
root
 |-- Species: string (nullable = true)
 |-- Weight: double (nullable = true)
 |-- Length1: double (nullable = true)
 |-- Length2: double (nullable = true)
 |-- Length3: double (nullable = true)
 |-- Height: double (nullable = true)
 |-- Width: double (nullable = true)
```

```
numeric_features = [t[0] for t in dataset.dtypes if t[1] == 'double']
dataset.select(numeric_features).describe().toPandas().transpose()
```

```
dataset.createOrReplaceTempView("df_sql")
fish = spark.sql("Select Species from df_sql")
fish = fish.rdd.map(lambda row : row.Species).collect()
pre = spark.sql("Select Weight from df_sql")
pre = pre.rdd.map(lambda row : row.Weight).collect()
plt.barh(fish,pre)
plt.xlabel("weight")
plt.ylabel("Species")


plt.show()
```



```
featureassembler = VectorAssembler(inputCols =
["Length1","Length2","Length3","Height","Width"], outputCol =
"Independent Features")

output = featureassembler.transform(dataset)
output.select("Independent Features").show()

+--------------------+
|Independent Features|
+--------------------+
|[23.2,25.4,30.0,1...|
|[24.0,26.3,31.2,1...|
|[23.9,26.5,31.1,1...|
|[26.3,29.0,33.5,1...|
|[26.5,29.0,34.0,1...|
|[26.8,29.7,34.7,1...|
|[26.8,29.7,34.5,1...|
|[27.6,30.0,35.0,1...|
|[27.6,30.0,35.1,1...|
|[28.5,30.7,36.2,1...|
|[28.4,31.0,36.2,1...|
|[28.7,31.0,36.2,1...|
|[29.1,31.5,36.4,1...|
|[29.5,32.0,37.3,1...|
|[29.4,32.0,37.2,1...|
|[29.4,32.0,37.2,1...|
|[30.4,33.0,38.3,1...|
|[30.4,33.0,38.5,1...|
|[30.9,33.5,38.6,1...|
|[31.0,33.5,38.7,1...|
+--------------------+
only showing top 20 rows
```

```
finalised_data = output.select("Independent Features", "Weight")
finalised_data
```

```
Out[46]: DataFrame[Independent Features: vector, Weight: double]
```

```
train_data, test_data = finalised_data.randomSplit([0.75, 0.25])
```

```
regressor = LinearRegression(featuresCol = 'Independent Features',
labelCol = 'Weight')
regressor = regressor.fit(train_data)
print("Coefficients: " + str(regressor.coefficients))
print("Intercept: " + str(regressor.intercept))
```

```
Coefficients: [60.56844095850656,-6.787366659094393,-
31.119499952522986,29.868185995482264,39.361152525390935]
Intercept: -477.8222167508448
```

```
pred_results = regressor.evaluate(test_data)
pred_results.predictions.show()
```

```
+--------------------+------+-------------------+
|Independent Features|Weight|         prediction|
+--------------------+------+-------------------+
|[9.3,9.8,10.8,1.7...|   6.7|-223.97296338856307|
|[10.7,11.2,12.4,2...|   9.8|-179.14673541674898|
|[11.4,12.0,13.2,2...|   9.8| -168.5240123377061|
|[12.9,14.1,16.2,4...|  40.0| -83.18666262216271|
|[13.2,14.3,15.2,2...|  19.7| -81.22183937364781|
|[13.8,15.0,16.2,2...|  19.9|  -86.3771578390747|
|[17.2,19.0,20.2,5...|  80.0|  94.77161223799664|
|[17.5,18.8,21.2,5...|  78.0|  75.64319706971452|
|[19.1,20.8,23.1,6...| 110.0| 136.87400653899198|
|[20.0,22.0,23.5,5...| 110.0| 175.11114853390717|
|[20.0,22.0,23.5,5...| 120.0| 160.12091870144258|
|[20.0,22.0,23.5,6...| 120.0| 169.53403069758582|
|[20.5,22.5,24.0,6...| 150.0| 209.75661024169648|
|[22.0,24.0,25.5,6...| 145.0| 239.20552985951764|
|[24.0,26.0,29.2,8...| 290.0|  332.7925786240622|
|[24.0,26.3,31.2,1...| 290.0|  368.6125641373547|
|[24.1,26.5,29.3,8...| 270.0| 300.72482398596304|
|[25.2,27.3,28.7,8...| 300.0|  420.8806978819236|
|[25.4,27.5,28.9,7...| 265.0| 355.85914840552397|
|[25.4,27.5,28.9,7...| 260.0| 359.31191070660174|
+--------------------+------+-------------------+
only showing top 20 rows
```

```
regressor.intercept
```

```
Out[50]: -477.8222167508448
```

```
train_data.describe().show()
```

```
+-------+------------------+
|summary|            Weight|
+-------+------------------+
|  count|               123|
|   mean| 392.2926829268293|
| stddev|338.09236786463646|
|    min|               0.0|
|    max|            1550.0|
+-------+------------------+
```