Ecommerce Revenue Prediction

Predicting spending of customer based on features like time spent on website, average time of sessions with personal stylists from the store etc., Hence helping business to make improvements in gaining more loyal customers.

Loading Data

```
import org.apache.spark.sql.Encoders

// defining the schema
case class Customer(Email: String,
                    Avatar: String,
                    Avg_Session_Length: Double,
                    Time_on_App: Double,
                    Time_on_Website: Double,
                    Length_of_Membership: Double,
                    Yearly_Amount_Spent: Double)

val CustomerSchema = Encoders.product[Customer].schema

val CustomerDF = spark.read.schema(CustomerSchema).option("header",
"true").csv("/FileStore/tables/ecommerce.csv")

display(CustomerDF)
```

	Email	Avatar	Avg_Session_Length	Time_on_App	Time_on

1	mstephenson@fernandez.com	Violet	34.49726773	12.65565115	39.57766
2	hduke@hotmail.com	DarkGreen	31.92627203	11.10946073	37.26895
3	pallen@yahoo.com	Bisque	33.00091476	11.33027806	37.11059
4	riverarebecca@gmail.com	SaddleBrown	34.30555663	13.71751367	36.72128
5	mstephens@davidson-herman.com	MediumAquaMarine	33.33067252	12.79518855	37.53665
6	alvareznancy@lucas.biz	FloralWhite	33.87103788	12.02692534	34.47687
7	katherine20@yahoo.com	DarkSlateBlue	32.0215955	11.36634831	36.68377

Showing all 500 rows.

```
CustomerDF.printSchema()
```

```
root
 |-- Email: string (nullable = true)
 |-- Avatar: string (nullable = true)
 |-- Avg_Session_Length: double (nullable = true)
 |-- Time_on_App: double (nullable = true)
 |-- Time_on_Website: double (nullable = true)
 |-- Length_of_Membership: double (nullable = true)
 |-- Yearly_Amount_Spent: double (nullable = true)
```

Data Summary

```
CustomerDF.select("Avg_Session_Length","Time_on_App", "Time_on_Website", "Length_of_Membership",
"Yearly_Amount_Spent").describe().show()
```

```
+-------+------------------+------------------+------------------+--------------------+-------------------+
|summary|Avg_Session_Length|       Time_on_App|   Time_on_Website|Length_of_Membership|Yearly_Amount_Spent|
+-------+------------------+------------------+------------------+--------------------+-------------------+
|  count|               500|               500|               500|                 500|                500|
```

```
|   mean|    33.05319351824|12.052487936928012|37.060445421080004|  3.5334615559298004|  499.3140382608002|
| stddev|0.9925631111602911|0.9942156084624618|1.0104889068105993|  0.9992775024372845|  79.31478155115914|
|    min|       29.53242897|       8.508152176|       33.91384725|          0.26990109|        256.6705823|
|    max|       36.13966249|       15.12699429|       40.00518164|         6.922689335|        765.5184619|
+-------+------------------+------------------+------------------+--------------------+-------------------+
```

Creating temporary view to query the dataframe

```
CustomerDF.createOrReplaceTempView("CustomerData")
```

```
%sql
select * from CustomerData
```

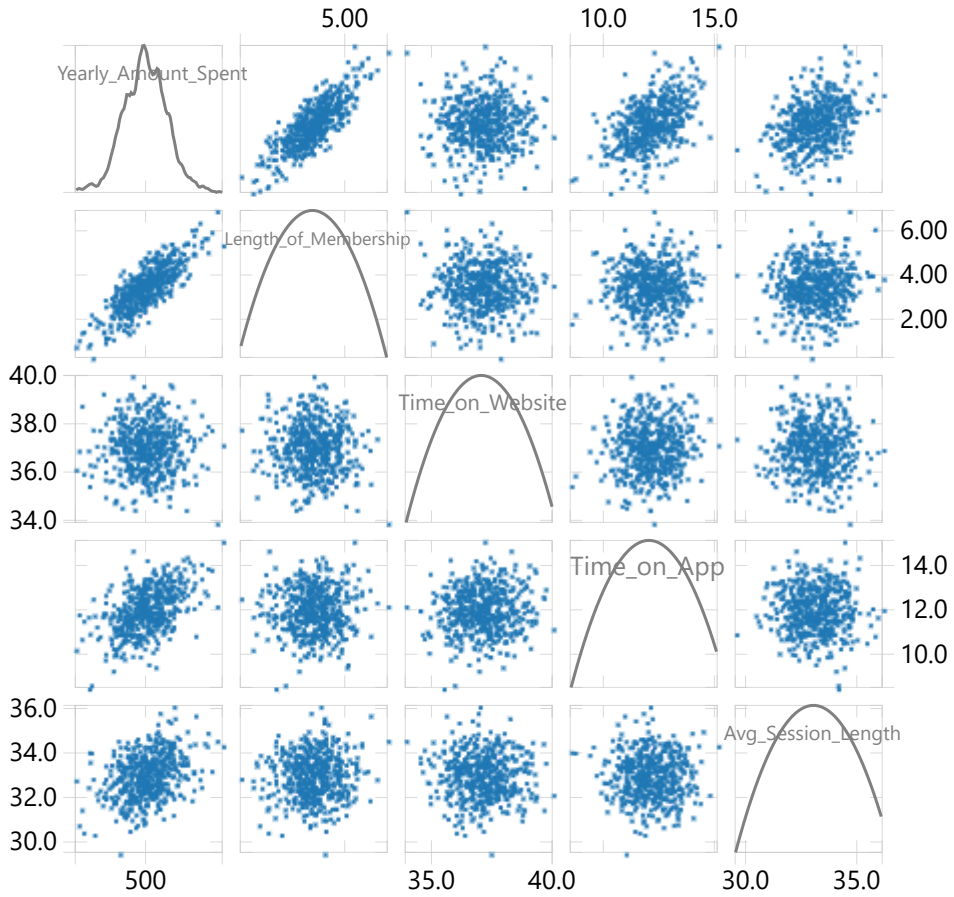
	Email	Avatar	Avg_Session_Length	Time_on_App	Time_on
1	mstephenson@fernandez.com	Violet	34.49726773	12.65565115	39.57766
2	hduke@hotmail.com	DarkGreen	31.92627203	11.10946073	37.26895
3	pallen@yahoo.com	Bisque	33.00091476	11.33027806	37.11059
4	riverarebecca@gmail.com	SaddleBrown	34.30555663	13.71751367	36.72128
5	mstephens@davidson-herman.com	MediumAquaMarine	33.33067252	12.79518855	37.53665
6	alvareznancy@lucas.biz	FloralWhite	33.87103788	12.02692534	34.47687
7	katherine20@yahoo.com	DarkSlateBlue	32.0215955	11.36634831	36.68377

Showing all 500 rows.

EDA

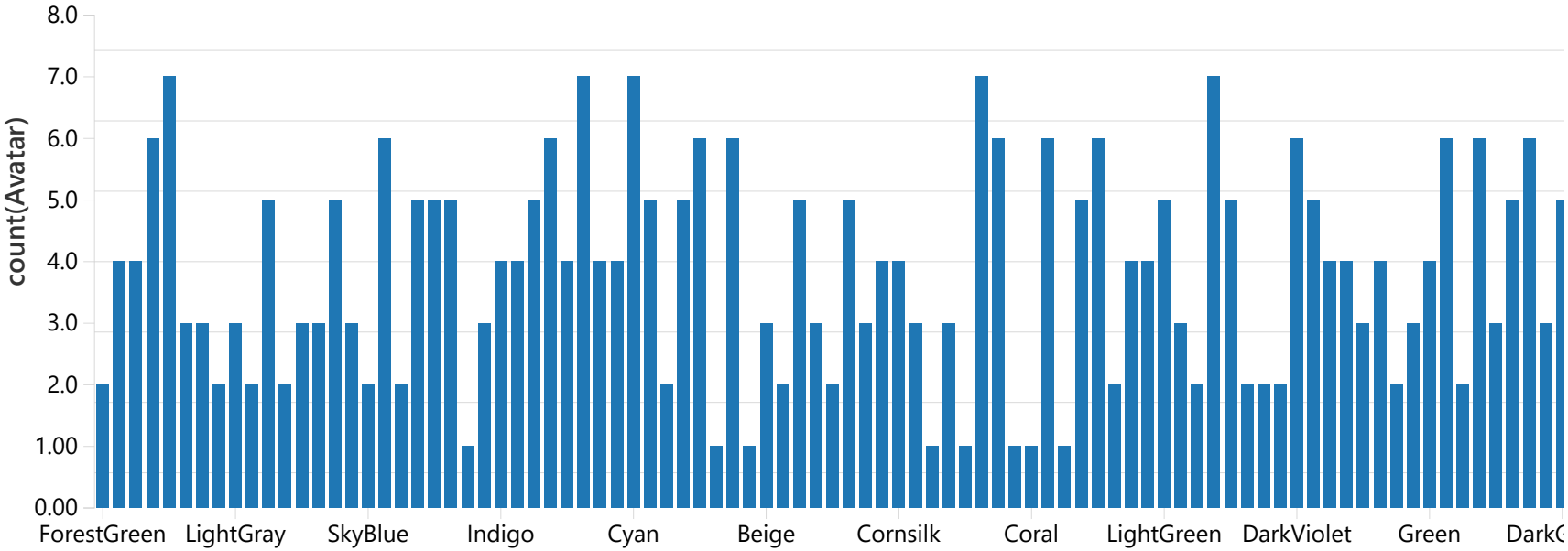
Relationship between all features

```
%sql
select Email, Avatar, Avg_Session_Length, Time_on_App, Time_on_Website, Length_of_Membership,
Yearly_Amount_Spent from CustomerData
```



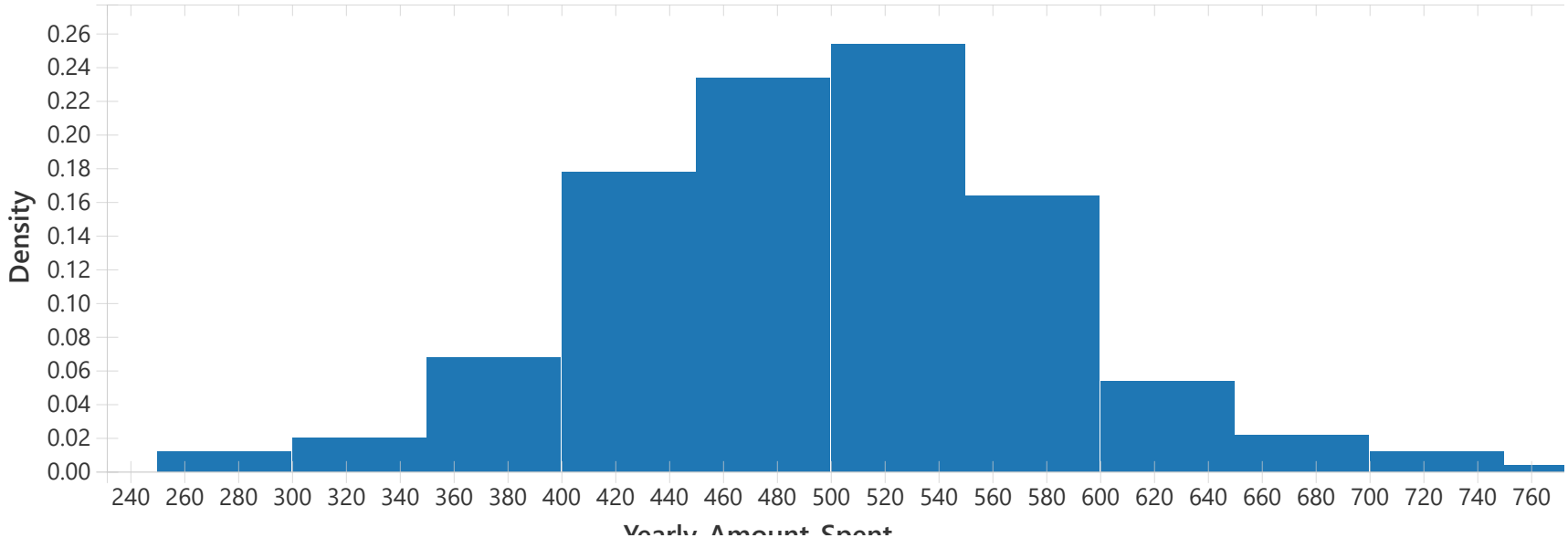
Types of Fashion

```
%sql

select Avatar as Fashion, count(Avatar) from CustomerData group by Avatar
```



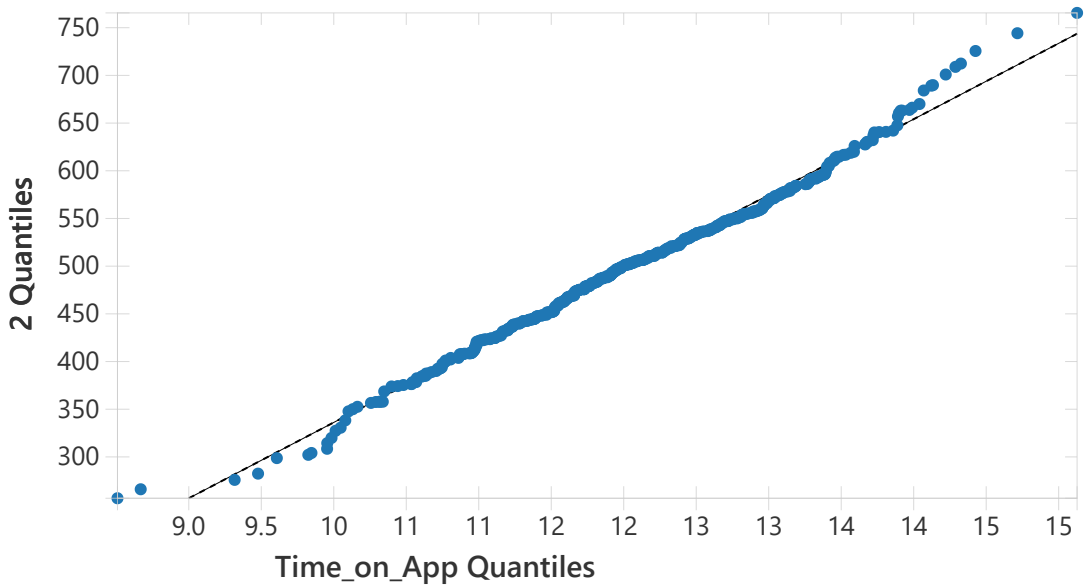
Amount spent on an yearly basis

```
%sql

select Yearly_Amount_Spent from CustomerData
```



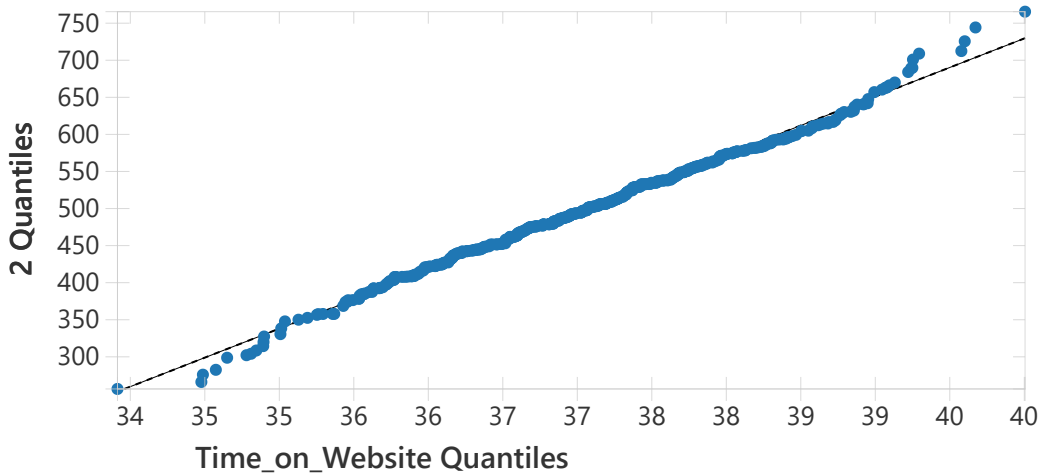
Amount spent based on time used on app

```
%sql

select Yearly_Amount_Spent, Time_on_App from CustomerData
```



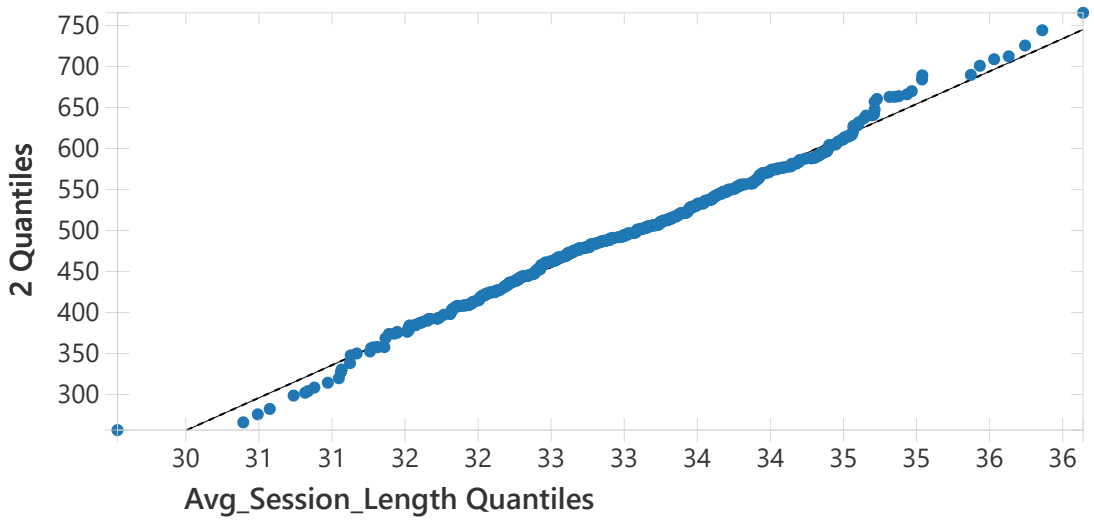
Amount spent based on time used on website

```
%sql

select Yearly_Amount_Spent, Time_on_Website from CustomerData
```



Amount spent based on average length of session with personal stylist

```
%sql

select Yearly_Amount_Spent, Avg_Session_Length from CustomerData
```



Linear Regression model

```
import org.apache.spark.sql.functions._
import org.apache.spark.sql.Row
import org.apache.spark.sql.types._

import org.apache.spark.ml.regression.LinearRegression
import org.apache.spark.ml.feature.VectorAssembler

import org.apache.spark.ml.attribute.Attribute
import org.apache.spark.ml.feature.{IndexToString, StringIndexer}
import org.apache.spark.ml.{Pipeline, PipelineModel}

//Concatenating columns into vector
var FeatureCol = Array("Email", "Avatar")

//Converting string values to indexes for categorical features
val index = FeatureCol.map { colName =>
  new StringIndexer().setInputCol(colName).setOutputCol(colName + "_indexed")
}

val lrpipeline = new Pipeline().setStages(index)

val FinaleCusotmerDF = lrpipeline.fit(CustomerDF).transform(CustomerDF)
```

```
import org.apache.spark.sql.functions._
import org.apache.spark.sql.Row
import org.apache.spark.sql.types._
import org.apache.spark.ml.regression.LinearRegression
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.attribute.Attribute
import org.apache.spark.ml.feature.{IndexToString, StringIndexer}
import org.apache.spark.ml.{Pipeline, PipelineModel}
FeatureCol: Array[String] = Array(Email, Avatar)
index: Array[org.apache.spark.ml.feature.StringIndexer] = Array(strIdx_666afd55fc21, strIdx_0e7ab34ea39d)
```

```
lrpipeline: org.apache.spark.ml.Pipeline = pipeline_6f4e33160fd8
FinaleCusotmerDF: org.apache.spark.sql.DataFrame = [Email: string, Avatar: string ... 7 more fields]
```

```
FinaleCusotmerDF.printSchema()
```

```
root
 |-- Email: string (nullable = true)
 |-- Avatar: string (nullable = true)
 |-- Avg_Session_Length: double (nullable = true)
 |-- Time_on_App: double (nullable = true)
 |-- Time_on_Website: double (nullable = true)
 |-- Length_of_Membership: double (nullable = true)
 |-- Yearly_Amount_Spent: double (nullable = true)
 |-- Email_indexed: double (nullable = false)
 |-- Avatar_indexed: double (nullable = false)
```

```
FinaleCusotmerDF.show()
```

```
+--------------------+---------------+------------------+-----------+---------------+--------------------+--
-----------------+-------------+--------------+
|               Email|          Avatar|Avg_Session_Length|Time_on_App|Time_on_Website|Length_of_Membership|Ye
arly_Amount_Spent|Email_indexed|Avatar_indexed|
+--------------------+---------------+------------------+-----------+---------------+--------------------+--
-----------------+-------------+--------------+
|mstephenson@ferna...|          Violet|       34.49726773|12.65565115|    39.57766802|         4.082620633|
587.951054|        342.0|          96.0|
|   hduke@hotmail.com|       DarkGreen|       31.92627203|11.10946073|    37.26895887|         2.664034182|
392.2049334|        190.0|          26.0|
|    pallen@yahoo.com|          Bisque|       33.00091476|11.33027806|    37.11059744|         4.104543202|
487.5475049|        355.0|           6.0|
|riverarebecca@gma...|     SaddleBrown|       34.30555663|13.71751367|    36.72128268|         3.120178783|
581.852344|        391.0|          18.0|
|mstephens@davidso...|MediumAquaMarine|       33.33067252|12.79518855|     37.5366533|         4.446308318|
```

```
599.406092|        341.0|          87.0|
|alvareznancy@luca...|     FloralWhite|       33.87103788|12.02692534|    34.47687763|         5.493507201|
637.1024479|         17.0|          49.0|
|katherine20@yahoo...|   DarkSlateBlue|        32.0215955|11.36634831|    36.68377615|         4.685017247|
521.5721748|        259.0|          80.0|
```

Splitting the data

```
val split = FinaleCusotmerDF.randomSplit(Array(0.7, 0.3))
val trainData = split(0)
val testData = split(1)
val trainRows = trainData.count()
val testRows = testData.count()
println("Training Rows: " + trainRows + " Testing Rows: " + testRows)
```

```
Training Rows: 343 Testing Rows: 157
split: Array[org.apache.spark.sql.Dataset[org.apache.spark.sql.Row]] = Array([Email: string, Avatar: string
... 7 more fields], [Email: string, Avatar: string ... 7 more fields])
trainData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Email: string, Avatar: string ... 7 more
fields]
testData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Email: string, Avatar: string ... 7 more
fields]
trainRows: Long = 343
testRows: Long = 157
```

Data Preproccessing

Combining categorical variables into a single feature using Vector Asse

```
val assembler = new VectorAssembler().setInputCols(Array("Email_indexed", "Avatar_indexed",
"Avg_Session_Length", "Time_on_App", "Time_on_Website", "Length_of_Membership")).setOutputCol("features")

val trainingData = assembler.transform(trainData).select($"features", $"Yearly_Amount_Spent".alias("label"))

trainingData.show()
```

```
+--------------------+-----------+
|            features|      label|
+--------------------+-----------+
|[0.0,68.0,33.7051...|521.2407802|
|[1.0,12.0,32.4495...|503.9783791|
|[2.0,101.0,33.452...|576.4776072|
|[3.0,18.0,31.4474...|418.6027421|
|[4.0,21.0,32.4256...|420.7376732|
|[5.0,69.0,33.5477...|476.1914133|
|[6.0,17.0,32.8487...|404.8245289|
|[7.0,65.0,32.6027...|482.1449969|
|[9.0,99.0,33.5030...|419.9387748|
|[10.0,1.0,32.1878...|452.3156755|
|[12.0,36.0,32.836...|256.6705823|
|[13.0,0.0,31.9673...|445.7498412|
|[14.0,48.0,32.887...| 684.163431|
|[15.0,105.0,33.63...|  497.81193|
|[16.0,72.0,31.954...|439.9978799|
|[17.0,49.0,33.871...|637.1024479|
|[22.0,76.0,32.959...| 448.340425|
|[24.0,44.0,34.501...| 584.105885|
```

```
val testingData = assembler.transform(testData).select($"features", $"Yearly_Amount_Spent".alias("trueLabel"))

testingData.show()
```

```
+--------------------+-----------+
|            features|  trueLabel|
+--------------------+-----------+
|[8.0,133.0,32.291...|494.5518611|
|[11.0,71.0,32.693...|501.9282649|
|[18.0,24.0,34.188...| 583.977802|
|[19.0,48.0,32.063...|378.3309069|
|[20.0,0.0,32.0961...|375.3984554|
|[21.0,2.0,33.9252...| 483.673308|
|[23.0,19.0,33.992...|492.6060127|
|[28.0,4.0,33.7801...|518.7864831|
|[30.0,34.0,33.700...|492.5568337|
|[31.0,26.0,32.351...|532.9352188|
|[33.0,41.0,33.566...|466.4211988|
|[39.0,7.0,34.3307...|558.4272572|
|[40.0,70.0,33.616...|611.0000251|
|[41.0,22.0,33.811...|535.3216101|
|[43.0,5.0,32.7391...|549.9041461|
|[44.0,83.0,33.066...|442.7228916|
|[46.0,100.0,33.35...|549.0082269|
|[49.0,51.0,33.540...|628.0478039|
```

Training the model

```
val lr = new
LinearRegression().setLabelCol("label").setFeaturesCol("features").setMaxIter(10).setRegParam(0.3)
val model = lr.fit(trainingData)
println("Model training complete")
```

```
Model training complete
lr: org.apache.spark.ml.regression.LinearRegression = linReg_eed5321c5154
model: org.apache.spark.ml.regression.LinearRegressionModel = LinearRegressionModel: uid=linReg_eed5321c5154,
numFeatures=6
```

Testing the model

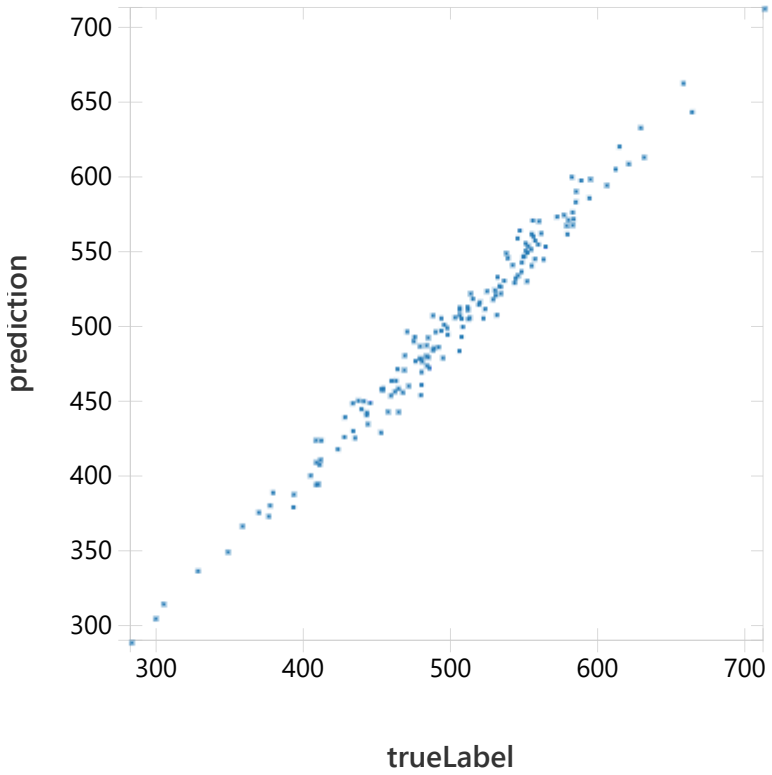
```
val prediction = model.transform(testingData)
val predicted = prediction.select("features", "trueLabel", "prediction" )
predicted.show()
```

```
+--------------------+-----------+------------------+
|            features|  trueLabel|        prediction|
+--------------------+-----------+------------------+
|[8.0,133.0,32.291...|494.5518611| 502.4519614738965|
|[11.0,71.0,32.693...|501.9282649|507.27083564989175|
|[18.0,24.0,34.188...| 583.977802| 584.3667815214767|
|[19.0,48.0,32.063...|378.3309069|390.35876463874433|
|[20.0,0.0,32.0961...|375.3984554| 374.5525249210964|
|[21.0,2.0,33.9252...| 483.673308|493.76242149757854|
|[23.0,19.0,33.992...|492.6060127|  506.717659805048|
|[28.0,4.0,33.7801...|518.7864831| 517.4332156350592|
|[30.0,34.0,33.700...|492.5568337| 498.4412885008537|
|[31.0,26.0,32.351...|532.9352188| 527.7380516044009|
|[33.0,41.0,33.566...|466.4211988|457.32176972292405|
|[39.0,7.0,34.3307...|558.4272572| 556.0535080061136|
|[40.0,70.0,33.616...|611.0000251| 606.2166181302844|
|[41.0,22.0,33.811...|535.3216101|  531.794104655874|
|[43.0,5.0,32.7391...|549.9041461| 556.7066550994368|
|[44.0,83.0,33.066...|442.7228916| 436.1210905358321|
|[46.0,100.0,33.35...|549.0082269| 548.1838317177062|
|[49.0,51.0,33.540...|628.0478039|   633.88168301012|
```

Model Performance

```
predicted.createOrReplaceTempView("CustomerData")
```

```
%sql

select prediction, trueLabel from CustomerData
```




```
Root Mean Square Error (RMSE): 9.950200403588989
import org.apache.spark.ml.evaluation.RegressionEvaluator
evaluation: org.apache.spark.ml.evaluation.RegressionEvaluator = RegressionEvaluator: uid=regEval_a758307e545
```

```
4, metricName=rmse, throughOrigin=false
rmse: Double = 9.950200403588989
```