Linear regression

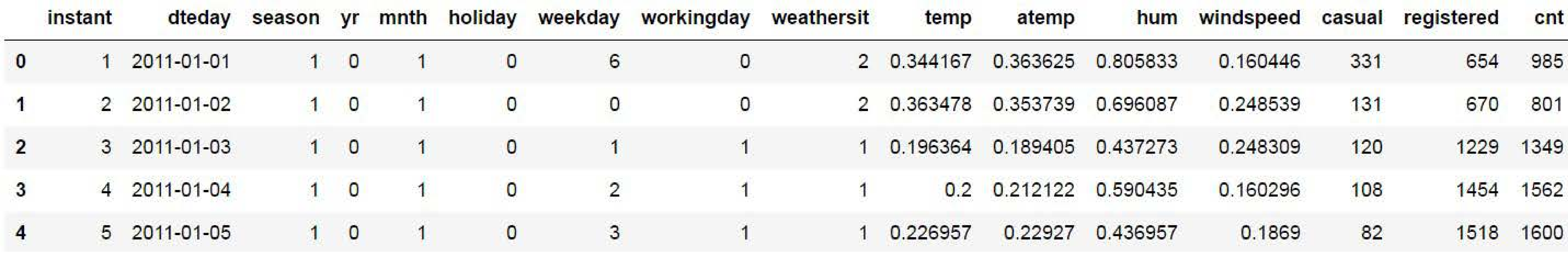
This week you will:

- how to prepare data for Spark MLlib
- how to make a predictions by linear regression
- how to estimate prediction quality

ENKEL
TOEGANKELIJK
VOOR FIETSERS
Fietsers
afstappen

```
bike_sharing = spark_session.read.csv(
    "/user/pmezentsev/regression_bike_sharing/day",
    header=True)
```

```
bike_sharing.limit(5).toPandas()
```

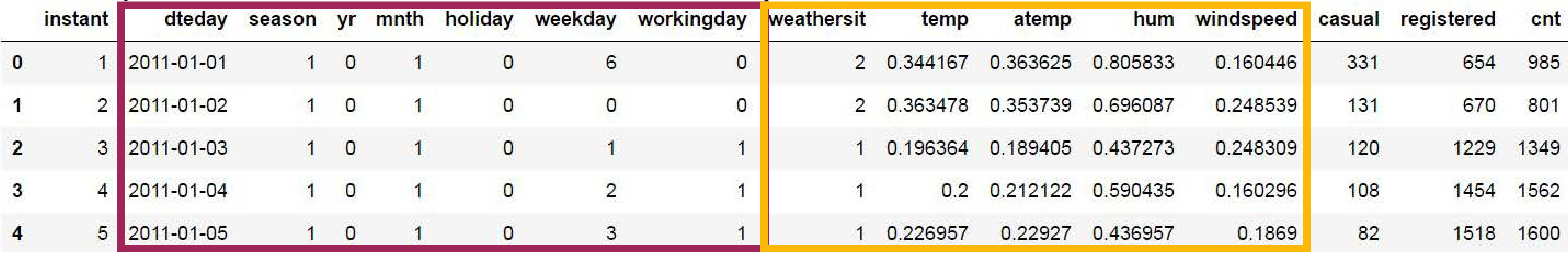


	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	4	2011-01-04	1	0	1	0	2	1	1	0.2	0.212122	0.590435	0.160296	108	1454	1562
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.22927	0.436957	0.1869	82	1518	1600

```
bike_sharing.limit(5).toPandas()
```

```
   instant      dteday  season  yr  mnth  holiday  weekday  workingday  weathersit      temp     atemp       hum  windspeed  casual  registered   cnt
0        1  2011-01-01       1   0     1        0        6           0           2  0.344167  0.363625  0.805833   0.160446     331         654   985
1        2  2011-01-02       1   0     1        0        0           0           2  0.363478  0.353739  0.696087   0.248539     131         670   801
2        3  2011-01-03       1   0     1        0        1           1           1  0.196364  0.189405  0.437273   0.248309     120        1229  1349
3        4  2011-01-04       1   0     1        0        2           1           1       0.2  0.212122  0.590435   0.160296     108        1454  1562
4        5  2011-01-05       1   0     1        0        3           1           1  0.226957   0.22927  0.436957     0.1869      82        1518  1600
```

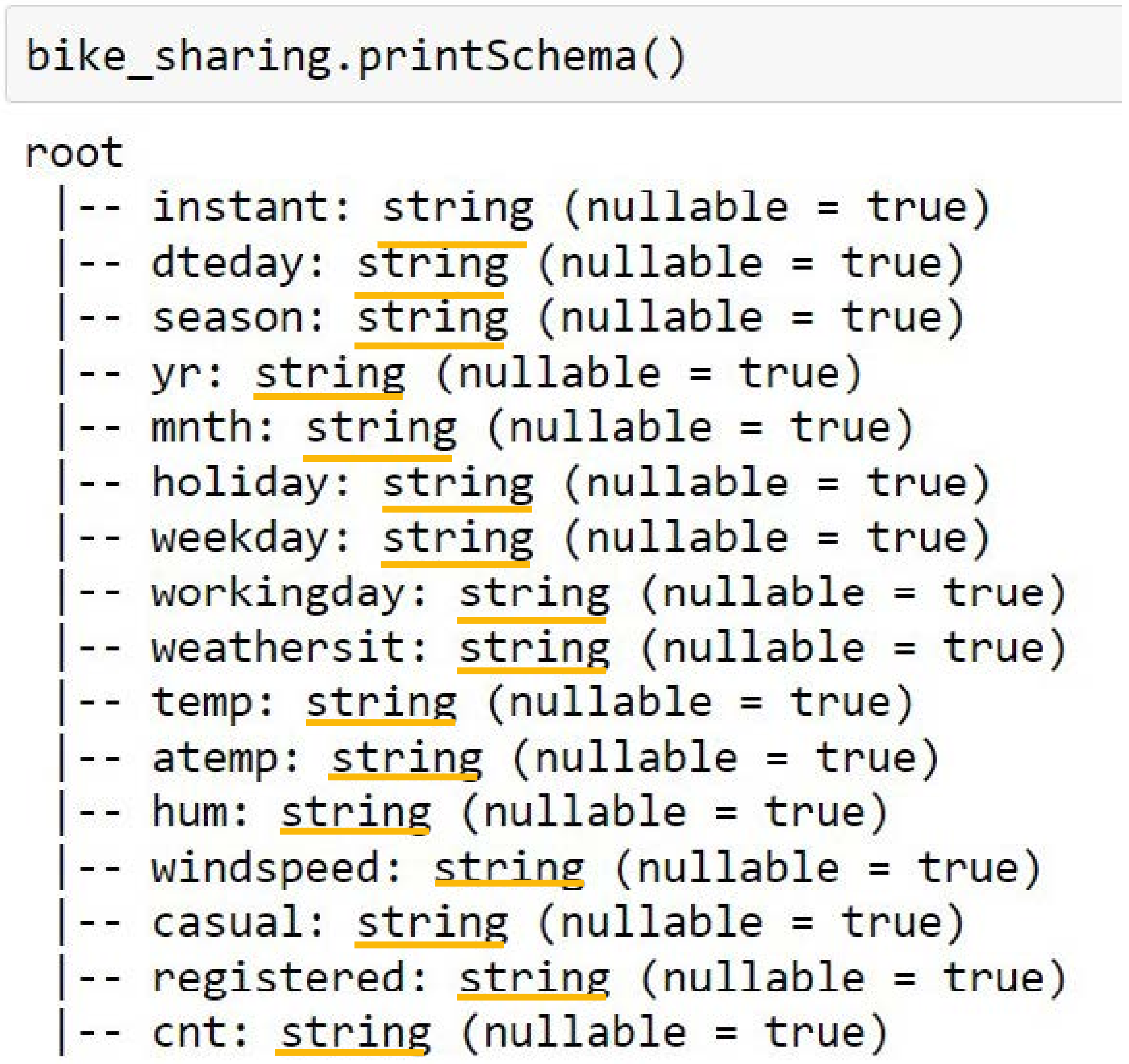
```
bike_sharing.limit(5).toPandas()
```



	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	4	2011-01-04	1	0	1	0	2	1	1	0.2	0.212122	0.590435	0.160296	108	1454	1562
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.22927	0.436957	0.1869	82	1518	1600

```
bike_sharing.limit(5).toPandas()
```

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	4	2011-01-04	1	0	1	0	2	1	1	0.2	0.212122	0.590435	0.160296	108	1454	1562
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.22927	0.436957	0.1869	82	1518	1600



```
bike_sharing.printSchema()
```

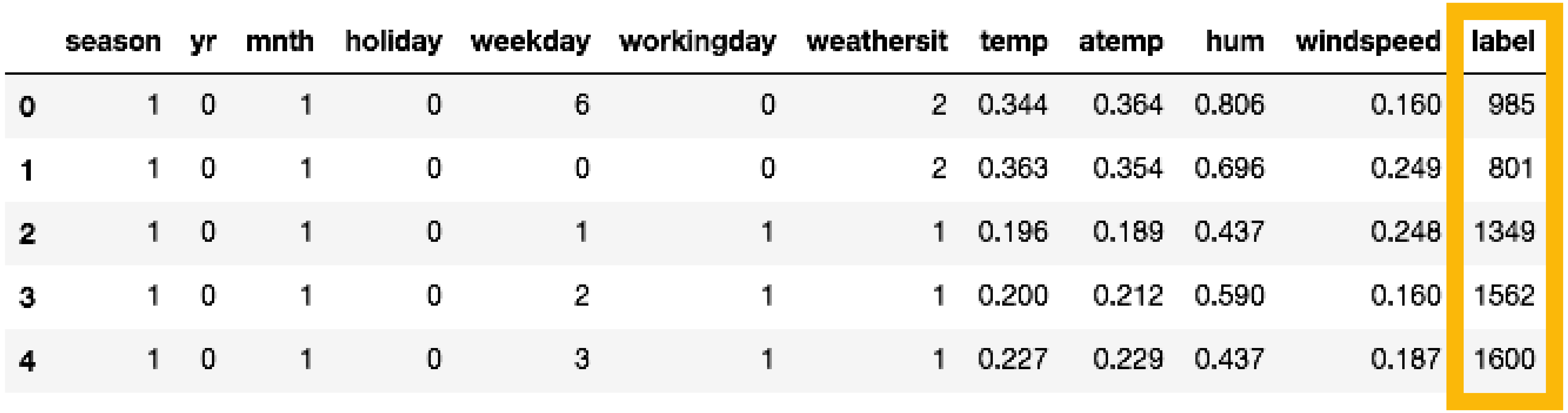
```
root
 |-- instant: string (nullable = true)
 |-- dteday: string (nullable = true)
 |-- season: string (nullable = true)
 |-- yr: string (nullable = true)
 |-- mnth: string (nullable = true)
 |-- holiday: string (nullable = true)
 |-- weekday: string (nullable = true)
 |-- workingday: string (nullable = true)
 |-- weathersit: string (nullable = true)
 |-- temp: string (nullable = true)
 |-- atemp: string (nullable = true)
 |-- hum: string (nullable = true)
 |-- windspeed: string (nullable = true)
 |-- casual: string (nullable = true)
 |-- registered: string (nullable = true)
 |-- cnt: string (nullable = true)
```

```
bike_sharing01 = bike_sharing.select(
    bike_sharing.season.astype("int"),
    bike_sharing.yr.astype("int"),
    bike_sharing.mnth.astype("int"),
    bike_sharing.holiday.astype("int"),
    bike_sharing.weekday.astype("int"),
    bike_sharing.workingday.astype("int"),
    bike_sharing.weathersit.astype("int"),
    bike_sharing.temp.astype("double"),
    bike_sharing.atemp.astype("double"),
    bike_sharing.hum.astype("double"),
    bike_sharing.windspeed.astype("double"),
    bike_sharing.cnt.astype("int").alias("label")
)
```

```
bike_sharing01 = bike_sharing.select(
    bike_sharing.season.astype("int"),
    bike_sharing.yr.astype("int"),
    bike_sharing.mnth.astype("int"),
    bike_sharing.holiday.astype("int"),
    bike_sharing.weekday.astype("int"),
    bike_sharing.workingday.astype("int"),
    bike_sharing.weathersit.astype("int"),
    bike_sharing.temp.astype("double"),
    bike_sharing.atemp.astype("double"),
    bike_sharing.hum.astype("double"),
    bike_sharing.windspeed.astype("double"),
    bike_sharing.cnt.astype("int").alias("label")
)
```

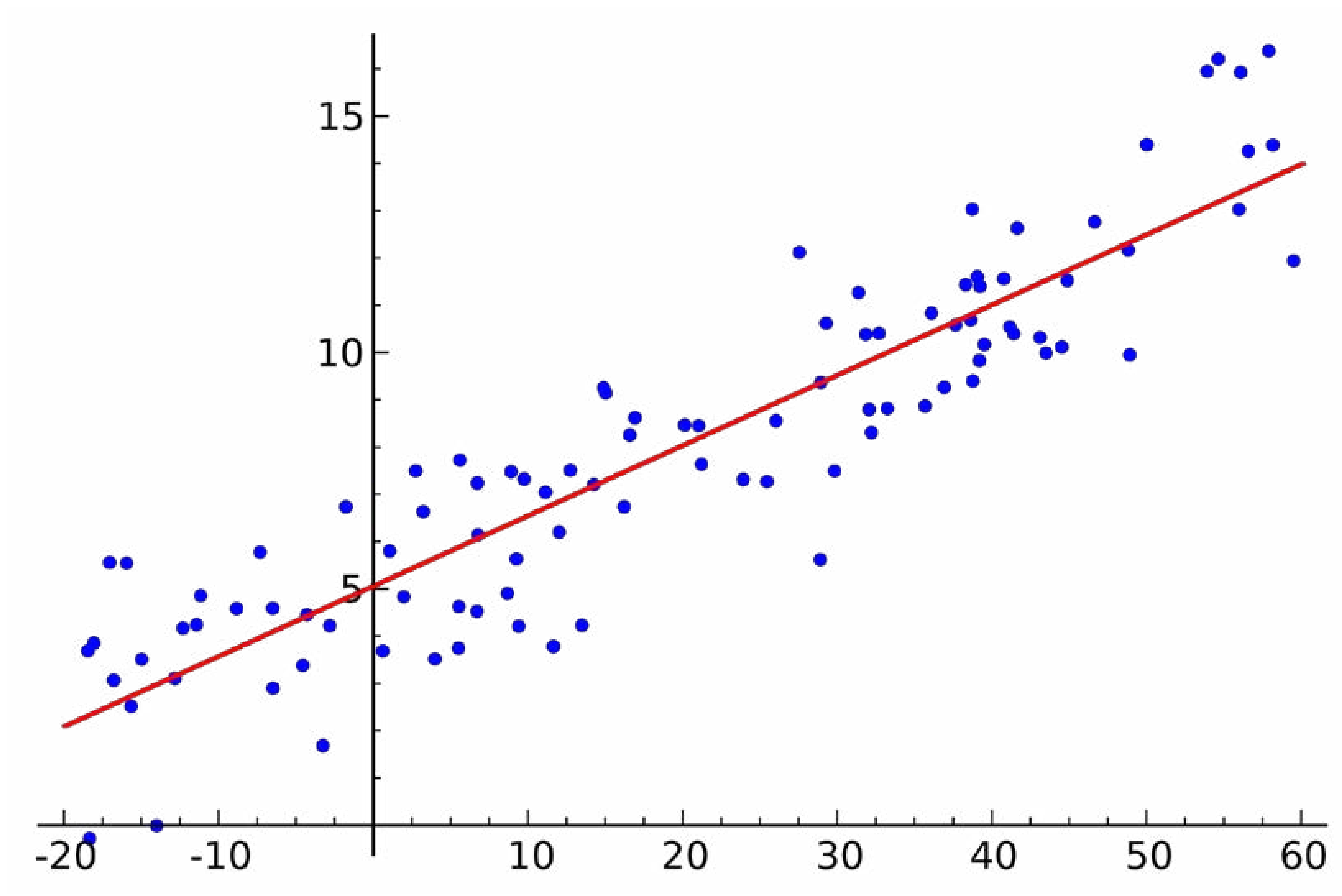
```
bike_sharing01 = bike_sharing.select(
    bike_sharing.season.astype("int"),
    bike_sharing.yr.astype("int"),
    bike_sharing.mnth.astype("int"),
    bike_sharing.holiday.astype("int"),
    bike_sharing.weekday.astype("int"),
    bike_sharing.workingday.astype("int"),
    bike_sharing.weathersit.astype("int"),
    bike_sharing.temp.astype("double"),
    bike_sharing.atemp.astype("double"),
    bike_sharing.hum.astype("double"),
    bike_sharing.windspeed.astype("double"),
    bike_sharing.cnt.astype("int").alias("label")
)
```

```
bike_sharing01.limit(5).toPandas()
```



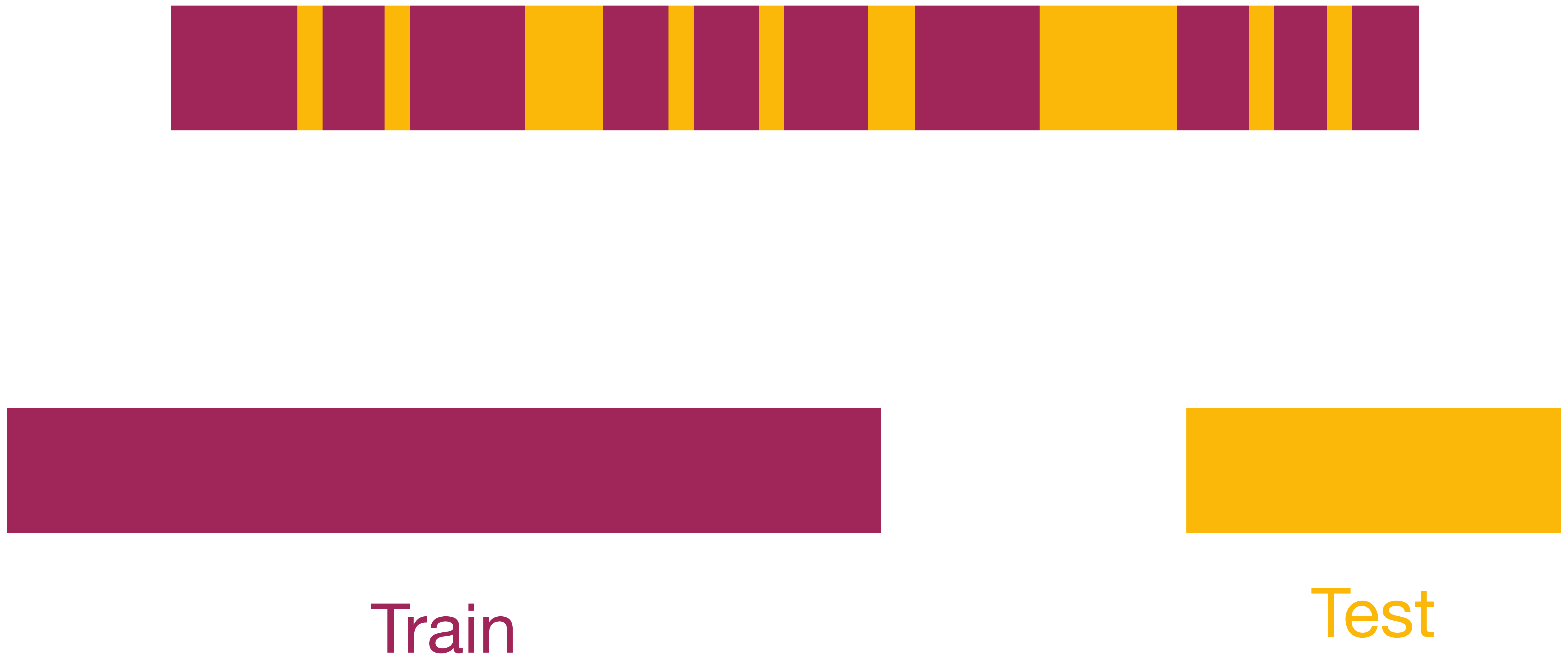
	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	label
0	1	0	1	0	6	0	2	0.344	0.364	0.806	0.160	985
1	1	0	1	0	0	0	2	0.363	0.354	0.696	0.249	801
2	1	0	1	0	1	1	1	0.196	0.189	0.437	0.248	1349
3	1	0	1	0	2	1	1	0.200	0.212	0.590	0.160	1562
4	1	0	1	0	3	1	1	0.227	0.229	0.437	0.187	1600

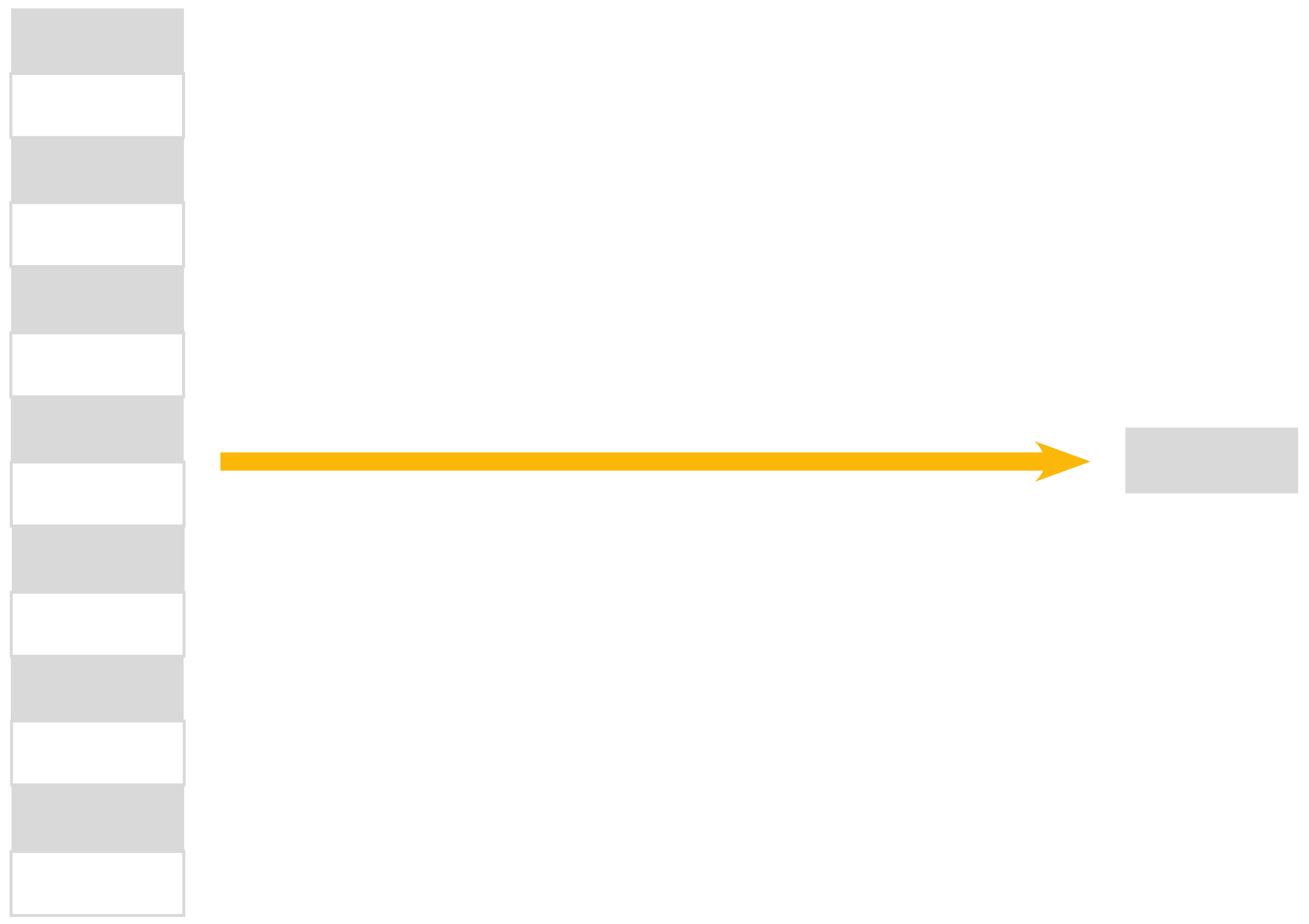
Linear regression



Train test splitting

```
train, test = bike_sharing.randomSplit([0.7,0.3])
```

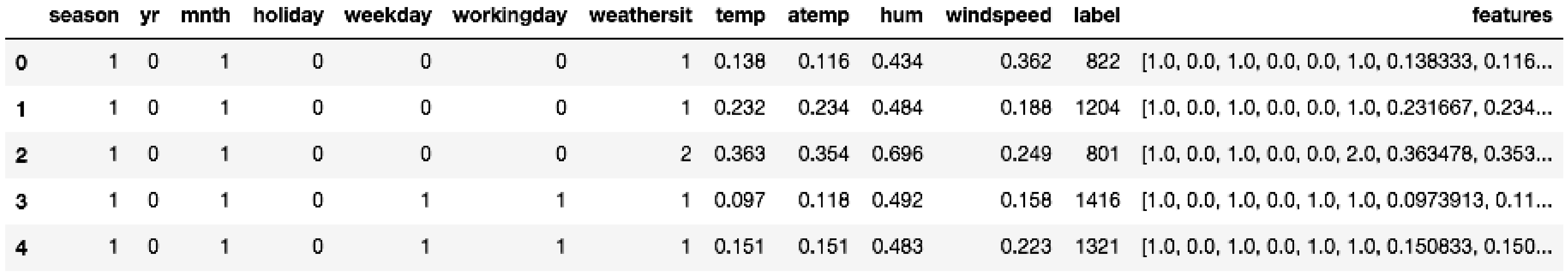




Vector

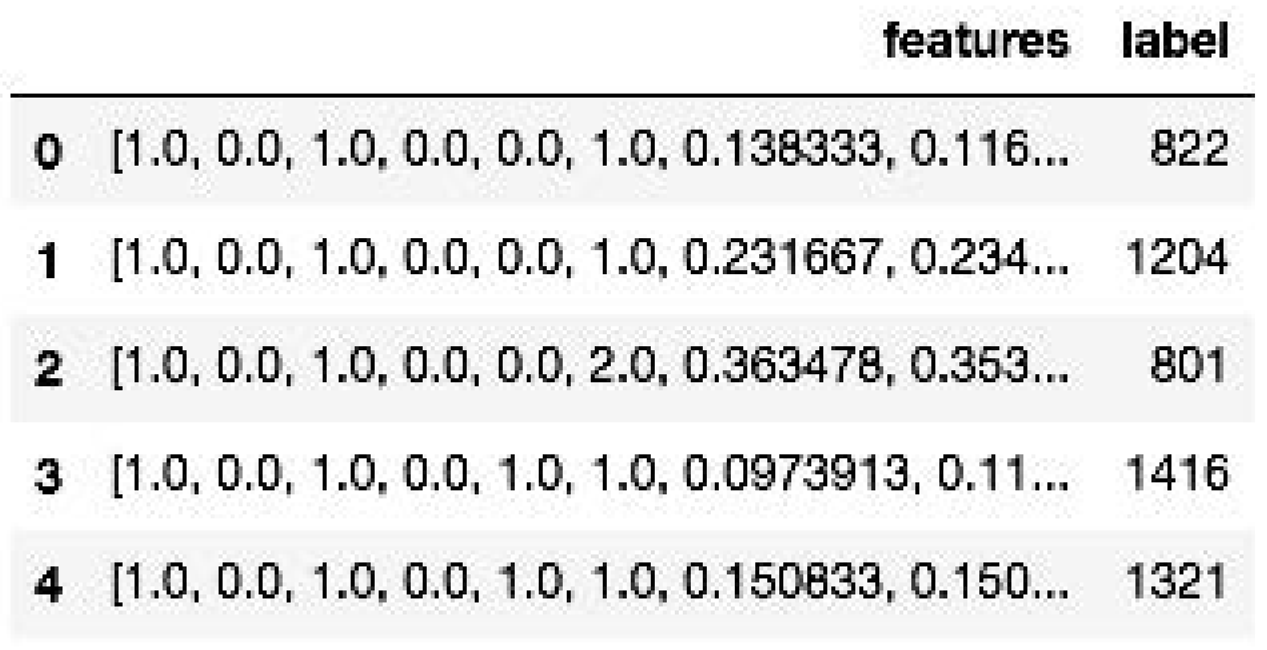
```python
from pyspark.ml.feature import VectorAssembler

assembler = VectorAssembler()\
  .setInputCols(["season",
                 "yr",
                 "mnth",
                 "holiday",
                 "weekday",
                 "weathersit",
                 "temp",
                 "atemp",
                 "hum",
                 "windspeed"])\
  .setOutputCol("features")

train01 = assembler.transform(train)
```

```
train01.limit(5).toPandas()
```



	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	label	features
0	1	0	1	0	0	0	1	0.138	0.116	0.434	0.362	822	[1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.138333, 0.116...
1	1	0	1	0	0	0	1	0.232	0.234	0.484	0.188	1204	[1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.231667, 0.234...
2	1	0	1	0	0	0	2	0.363	0.354	0.696	0.249	801	[1.0, 0.0, 1.0, 0.0, 0.0, 2.0, 0.363478, 0.353...
3	1	0	1	0	1	1	1	0.097	0.118	0.492	0.158	1416	[1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0973913, 0.11...
4	1	0	1	0	1	1	1	0.151	0.151	0.483	0.223	1321	[1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.150833, 0.150...

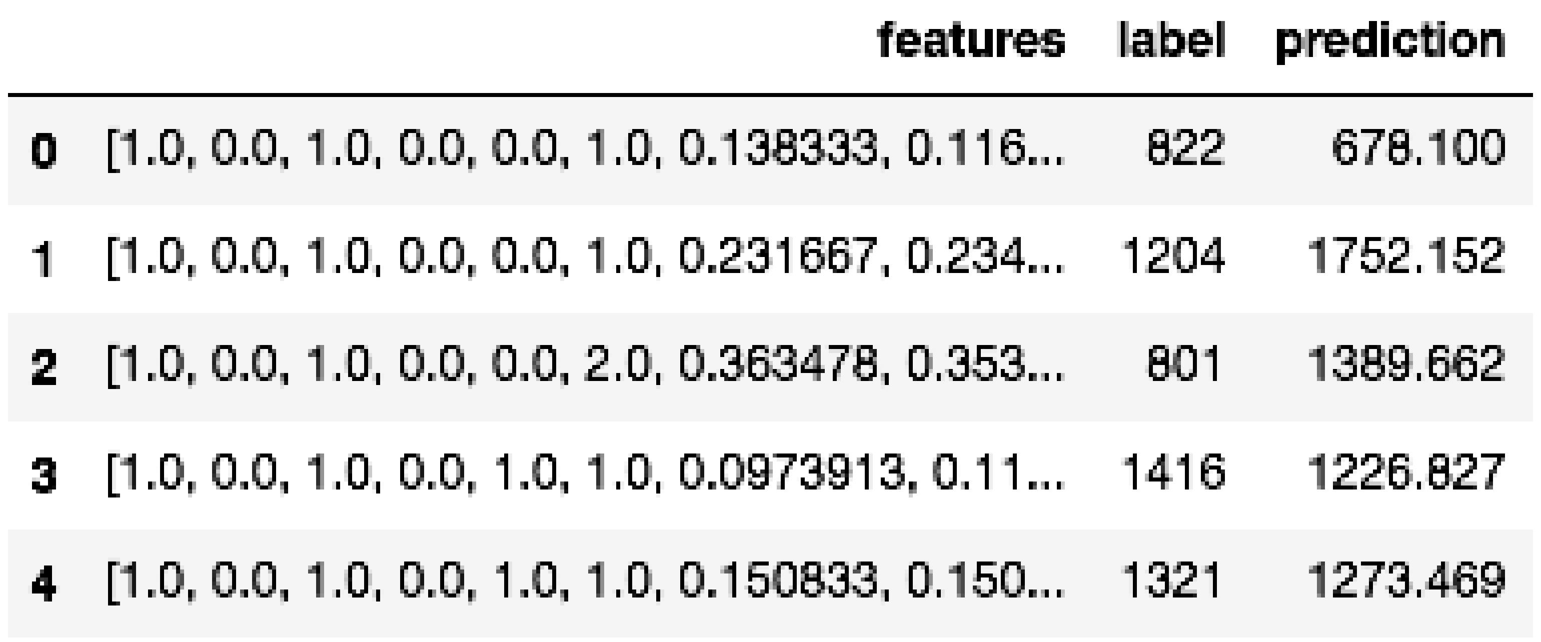
```
train02 = train01.select("features","label")
train02.limit(5).toPandas()
```



	features	label
0	[1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.138333, 0.116...	822
1	[1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.231667, 0.234...	1204
2	[1.0, 0.0, 1.0, 0.0, 0.0, 2.0, 0.363478, 0.353...	801
3	[1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0973913, 0.11...	1416
4	[1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.150833, 0.150...	1321

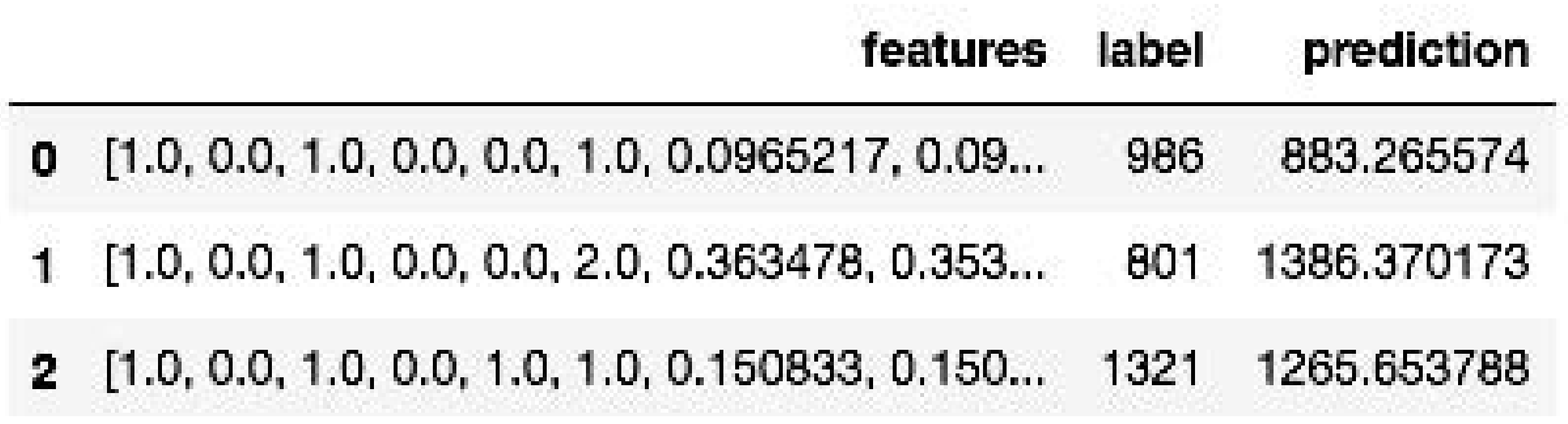
```python
from pyspark.ml.regression import LinearRegression
lr = LinearRegression()
model = lr.fit(train03)
```

```
train03 = model.transform(train03)
train03.limit(5).toPandas()
```



	features	label	prediction
0	[1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.138333, 0.116...	822	678.100
1	[1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.231667, 0.234...	1204	1752.152
2	[1.0, 0.0, 1.0, 0.0, 0.0, 2.0, 0.363478, 0.353...	801	1389.662
3	[1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0973913, 0.11...	1416	1226.827
4	[1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.150833, 0.150...	1321	1273.469

```
test01 = assembler.transform(test01)
test02 = test02.select("features","label")
test03 = model.transform(test03)

test03.limit(3).toPandas()
```



	features	label	prediction
0	[1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0965217, 0.09...	986	883.265574
1	[1.0, 0.0, 1.0, 0.0, 0.0, 2.0, 0.363478, 0.353...	801	1386.370173
2	[1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.150833, 0.150...	1321	1265.653788

```python
from pyspark.ml.evaluation import RegressionEvaluator
evaluator = RegressionEvaluator()
```

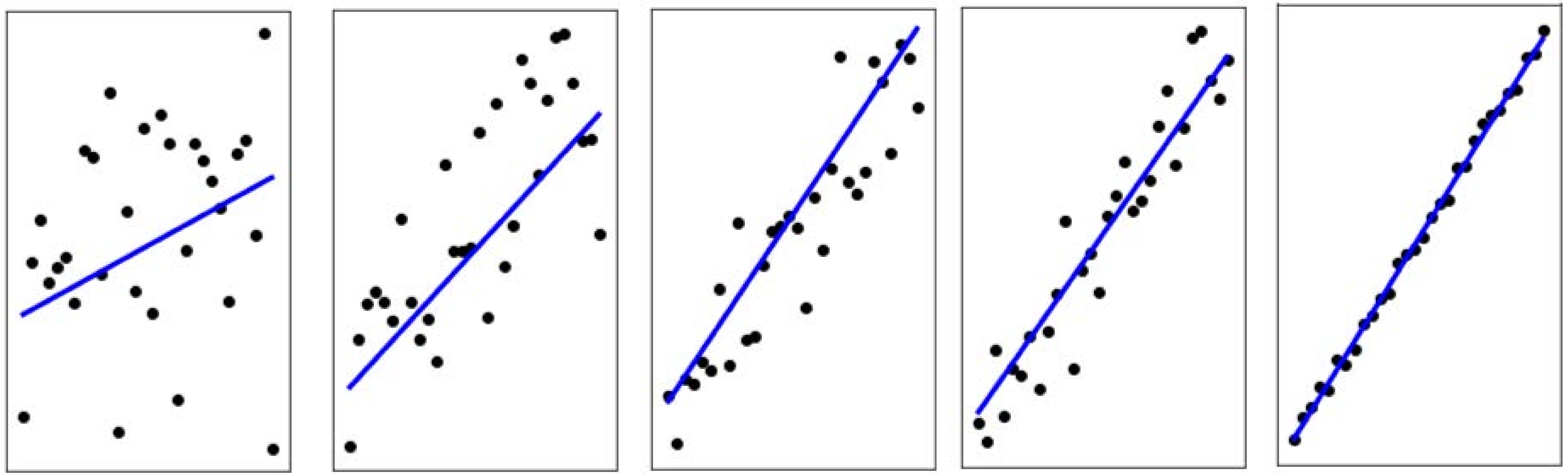
Coefficient of Determination

$$R^2$$

```
evaluator.evaluate(test04,
                   {evaluator.metricName: "r2"})
```

```
0.762
```

0.76



Root mean-square Error

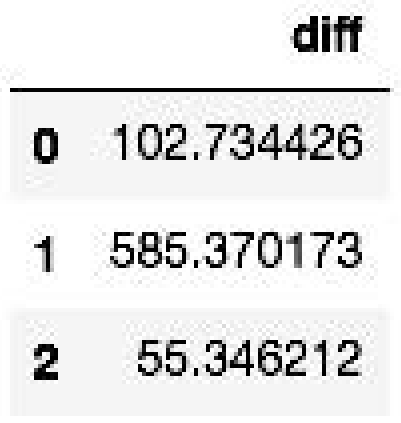
$$\mathrm{RMSE} = \sqrt{\frac{\Sigma err_i^2}{n}}$$

```
evaluator.evaluate(test04,
                   {evaluator.metricName: "rmse"})
```

```
912.341
```

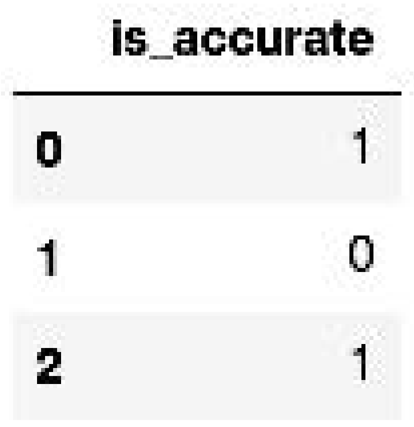
How often did we miss less then on 300 bikes?

```
test03.select(f.abs(f.col("label")-f.col("prediction")).alias("diff"))\
      .limit(3).toPandas()
```



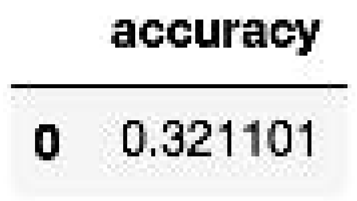
```
         diff
0  102.734426
1  585.370173
2   55.346212
```

```
test04.select(f.abs(f.col("label")-f.col("prediction")).alias("diff"))\
      .select(f.when(f.col("diff")<300, 1).otherwise(0).alias("is_accurate"))\
      .limit(3).toPandas()
```



	is_accurate
0	1
1	0
2	1

```python
test03.select(f.abs(f.col("label")-f.col("prediction")).alias("diff"))\
      .select(f.when(f.col("diff")<300, 1).otherwise(0).alias("is_accurate"))\
      .agg(f.mean("is_accurate").alias("accuracy"))\
      .toPandas()
```



```
   accuracy
0  0.321101
```

You have learned

- How to prepare data for Spark MLlib
- How to make a predictions by linear regression
- How to estimate prediction quality