LabProject_Altendorfer_Eckmayr

January 22, 2024

1 LabProject

Altendorfer, Eckmayr

2024-01-19 Link zum Repository: GitHub

Link zum Notebook: nbviewer

Link zum Datensatz: Kaggle

[1]:
```
import plotly.express as px
import plotly.io as pio
import plotly.graph_objects as go
import matplotlib.pyplot as plt

import warnings

from scipy import stats

from plotly.subplots import make_subplots

from pyspark.sql import SparkSession
from pyspark.sql import functions as F
from pyspark.sql.types import IntegerType

from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler
from pyspark.ml.regression import LinearRegression
from pyspark.ml.evaluation import RegressionEvaluator

pio.renderers.default='notebook+pdf+svg'

warnings.filterwarnings('ignore')
warnings.simplefilter('ignore')
```

[2]:
```
spark = SparkSession.builder \
    .master('local[*]') \
    .appName('LabProject_Altendorfer-Eckmayr') \
    .getOrCreate()
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
24/01/22 10:22:40 WARN NativeCodeLoader: Unable to load native-hadoop library
for your platform… using builtin-java classes where applicable
```

[3]:
```
spark
```

[3]:
```
<pyspark.sql.session.SparkSession at 0x7f57ac27f210>
```

1.1 Dataset Overview

[4]:
```
students_data = spark.read.csv('StudentsPerformance.csv', header=True,␣
↪inferSchema=True)
```

[5]:
```
students_data.show(5)
```

```
+------+--------------+--------------------------+------------+---------------
-------+----------+-------------+-------------+
|gender|race/ethnicity|parental level of education|       lunch|test preparation
course|math score|reading score|writing score|
+------+--------------+--------------------------+------------+---------------
-------+----------+-------------+-------------+
|female|       group B|          bachelor's degree|    standard|
none|        72|           72|           74|
|female|       group C|               some college|    standard|
completed|        69|           90|           88|
|female|       group B|           master's degree|    standard|
none|        90|           95|           93|
|  male|       group A|         associate's degree|free/reduced|
none|        47|           57|           44|
|  male|       group C|               some college|    standard|
none|        76|           78|           75|
+------+--------------+--------------------------+------------+---------------
-------+----------+-------------+-------------+
only showing top 5 rows
```

[6]:
```
students_data.printSchema()
```

```
root
 |-- gender: string (nullable = true)
 |-- race/ethnicity: string (nullable = true)
 |-- parental level of education: string (nullable = true)
 |-- lunch: string (nullable = true)
 |-- test preparation course: string (nullable = true)
 |-- math score: integer (nullable = true)
 |-- reading score: integer (nullable = true)
```

```
 |-- writing score: integer (nullable = true)
```

[7]:
```
students_data.describe().show()
```

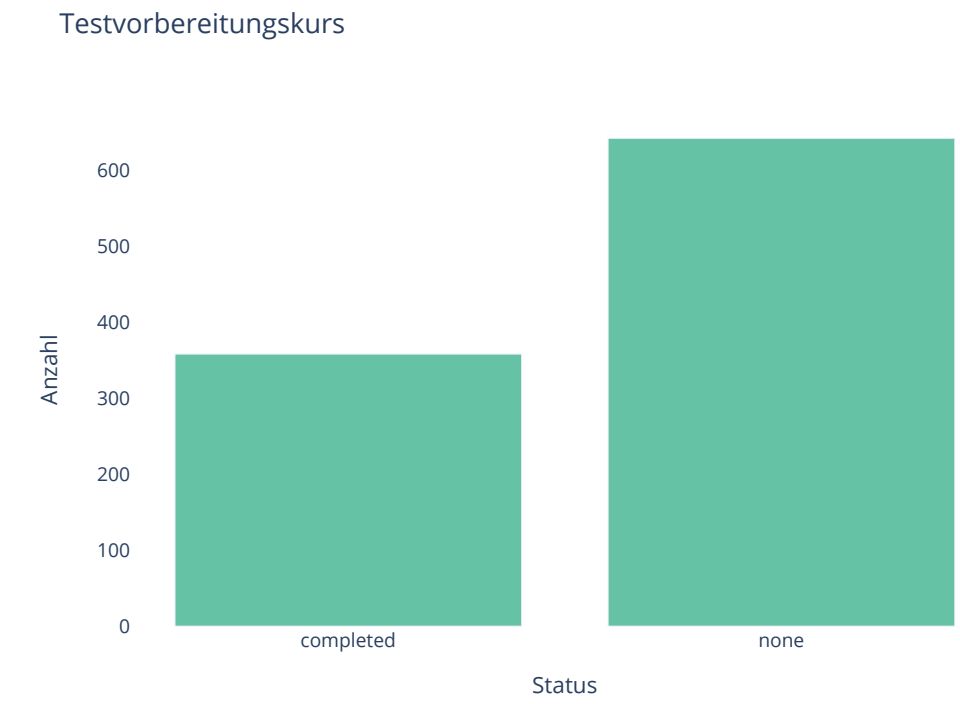
```
24/01/22 10:22:46 WARN package: Truncated the string representation of a plan
since it was too large. This behavior can be adjusted by setting
'spark.sql.debug.maxToStringFields'.
[Stage 3:>                                                          (0 + 1) / 1]
+-------+------+--------------+------------------------+-----------+--------
--------------+------------------+------------------+-----------------+
|summary|gender|race/ethnicity|parental level of education|       lunch|test
preparation course|        math score|     reading score|    writing score|
+-------+------+--------------+------------------------+-----------+--------
--------------+------------------+------------------+-----------------+
|  count|  1000|          1000|                    1000|       1000|
1000|              1000|              1000|             1000|
|   mean|  null|          null|                    null|       null|
null|            66.089|            69.169|           68.054|
| stddev|  null|          null|                    null|       null|
null|15.163080096009454|14.600191937252223|15.19565701086966|
|    min|female|       group A|         associate's degree|free/reduced|
completed|                 0|                17|               10|
|    max|  male|       group E|           some high school|   standard|
none|               100|               100|              100|
+-------+------+--------------+------------------------+-----------+--------
--------------+------------------+------------------+-----------------+
```

[8]:
```
students_data.groupBy('test preparation course').count().show()
```

```
+----------------------+-----+
|test preparation course|count|
+----------------------+-----+
|             completed|  358|
|                  none|  642|
+----------------------+-----+
```

[9]:
```
df = students_data.groupBy('test preparation course').count().toPandas()

fig = px.bar(df,
             x='test preparation course',
             y=['count'],
             title='Testvorbereitungskurs',
```

```
            labels={'test preparation course': 'Status', 'value': 'Anzahl',␣
 ↪'variable': 'Status'},
            color_discrete_sequence=px.colors.qualitative.Set2)

fig.update_layout(showlegend=False, paper_bgcolor="white", plot_bgcolor="white")
fig.show()
```



Loading [MathJax]/extensions/MathMenu.js

1.2 Forschungsfrage 1

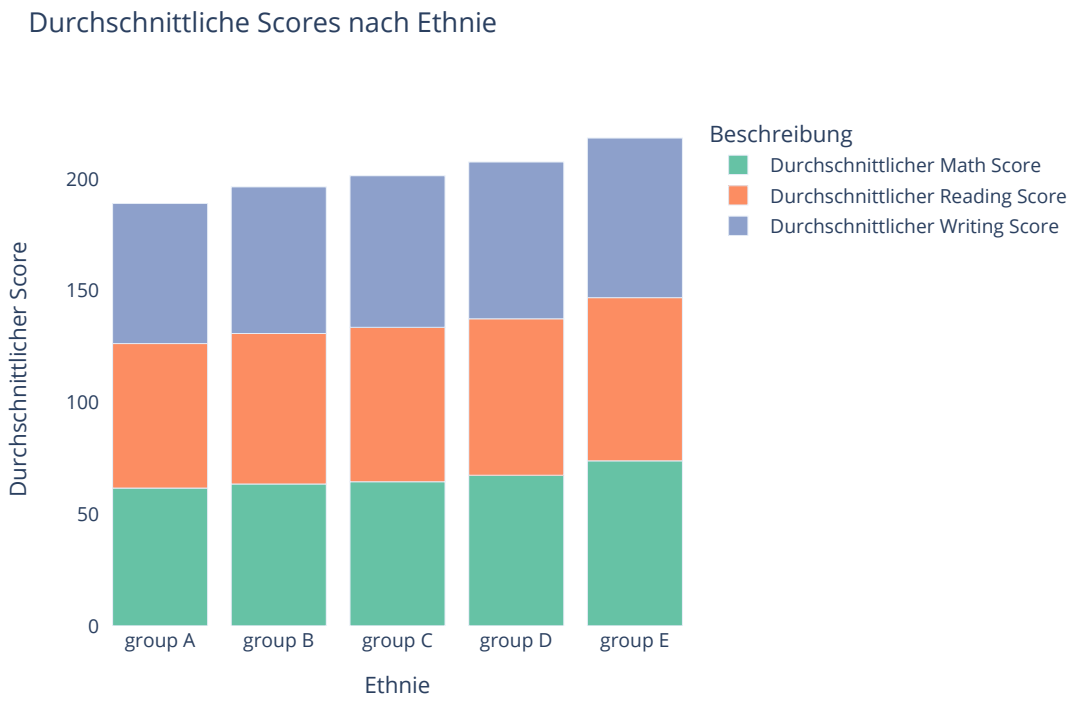
Kann ein Unterschied zwischen den ethnischen Gruppen in den erzielten Prüfungsergebnissen beobachtet werden?

[10]:
```
average_scores = students_data.groupBy("race/ethnicity").agg(
    F.avg("math score").alias("Durchschnittlicher Math Score"),
    F.avg("reading score").alias("Durchschnittlicher Reading Score"),
    F.avg("writing score").alias("Durchschnittlicher Writing Score")
)

average_scores.show()
```

```
+--------------+----------------------------+--------------------------------+-
--------------------------------+
```

```
|race/ethnicity|Durchschnittlicher Math Score|Durchschnittlicher Reading
Score|Durchschnittlicher Writing Score|
+--------------+----------------------------+-------------------------------+-
------------------------------+
|       group B|           63.45263157894737|              67.35263157894737|
65.6|
|       group C|           64.46394984326018|              69.10344827586206|
67.82758620689656|
|       group D|           67.36259541984732|              70.03053435114504|
70.14503816793894|
|       group A|           61.62921348314607|              64.67415730337079|
62.674157303370784|
|       group E|           73.82142857142857|              73.02857142857142|
71.40714285714286|
+--------------+----------------------------+-------------------------------+-
------------------------------+
```

[11]:
```
#df = average_scores.toPandas().set_index(["race/ethnicity"])\
#                              .stack()\
#                              .reset_index()\
#                              .rename(columns={'level_1':'desc', 0: 'score'})\
#                              .sort_values('race/ethnicity')

df = average_scores.toPandas()\
                   .sort_values('race/ethnicity')

fig = px.bar(df,
             x='race/ethnicity',
             y=['Durchschnittlicher Math Score', 'Durchschnittlicher Reading␣
 ↪Score', 'Durchschnittlicher Writing Score'],
             #y='score',
             #color='desc',
             barmode='stack',
             title='Durchschnittliche Scores nach Ethnie',
             labels={'race/ethnicity': 'Ethnie', 'variable': 'Beschreibung',␣
 ↪'value': 'Durchschnittlicher Score'},
             color_discrete_sequence=px.colors.qualitative.Set2)

fig.update_layout(paper_bgcolor="white", plot_bgcolor="white")
fig.show()
```



1.3 Forschungsfrage 2

Beeinflusst das Bildungsniveau der Eltern die Wahrscheinlichkeit einen Vorbereitungskurs zu absolvieren und wirkt sich die Absolvierung eines solchen auf die Prüfungsergebnisse aus?

1.3.1 Teil 1

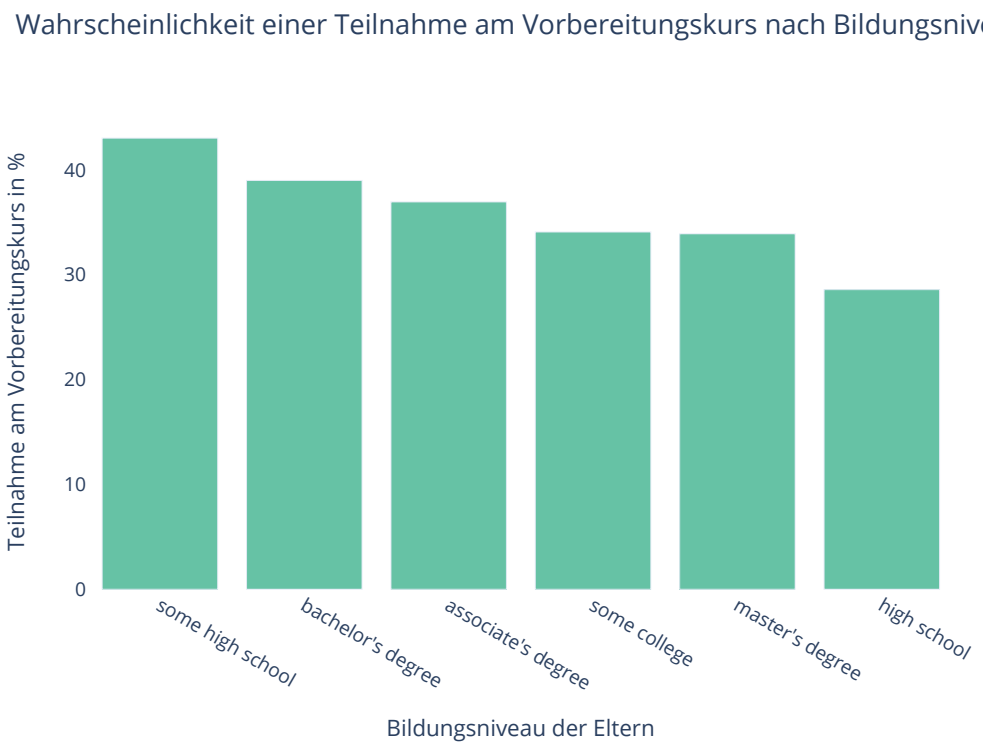
Beeinflusst das Bildungsniveau der Eltern die Wahrscheinlichkeit einen Vorbereitungskurs zu absolvieren?

[12]:
```
total = students_data.groupBy("parental level of education").count().
 ↪withColumnRenamed("count", "total")
preperation_count = students_data.groupBy("parental level of education", "test␣
 ↪preparation course").count().withColumnRenamed("count", "preperation_count")

prep_percentage = preperation_count.join(total, "parental level of education")
prep_percentage = prep_percentage.withColumn(
    "percentage",
    F.col("preperation_count") / F.col("total") * 100
)
```

```
filtered_prep_percentage = prep_percentage.filter(F.col("test preparation␣
 ↪course") == "completed")
sorted_prep_percentage = filtered_prep_percentage.sort("percentage", ascending␣
 ↪= False)
sorted_prep_percentage.show()
```

```
+--------------------------+----------------------+----------------+-----+---
---------------+
|parental level of education|test preparation course|preperation_count|total|
percentage|
+--------------------------+----------------------+----------------+-----+---
---------------+
|           some high school|             completed|              77|  179|
43.01675977653631|
|          bachelor's degree|             completed|              46|
118|38.983050847457626|
|         associate's degree|             completed|              82|  222|
36.93693693693694|
|               some college|             completed|              77|
226|34.070796460176986|
|            master's degree|             completed|              20|   59|
33.89830508474576|
|                high school|             completed|              56|  196|
28.57142857142857|
+--------------------------+----------------------+----------------+-----+---
---------------+
```

[13]:
```
df = sorted_prep_percentage.toPandas()

fig = px.bar(df,
             x='parental level of education',
             y=['percentage'],
             title='Wahrscheinlichkeit einer Teilnahme am Vorbereitungskurs␣
 ↪nach Bildungsniveau der Eltern',
             labels={'parental level of education': 'Bildungsniveau der␣
 ↪Eltern', 'value': 'Teilnahme am Vorbereitungskurs in %', 'variable':␣
 ↪'Beschreibung'},
             color_discrete_sequence=px.colors.qualitative.Set2)

fig.update_layout(showlegend=False, paper_bgcolor="white", plot_bgcolor="white")
fig.show()
```



1.3.2 Teil 2

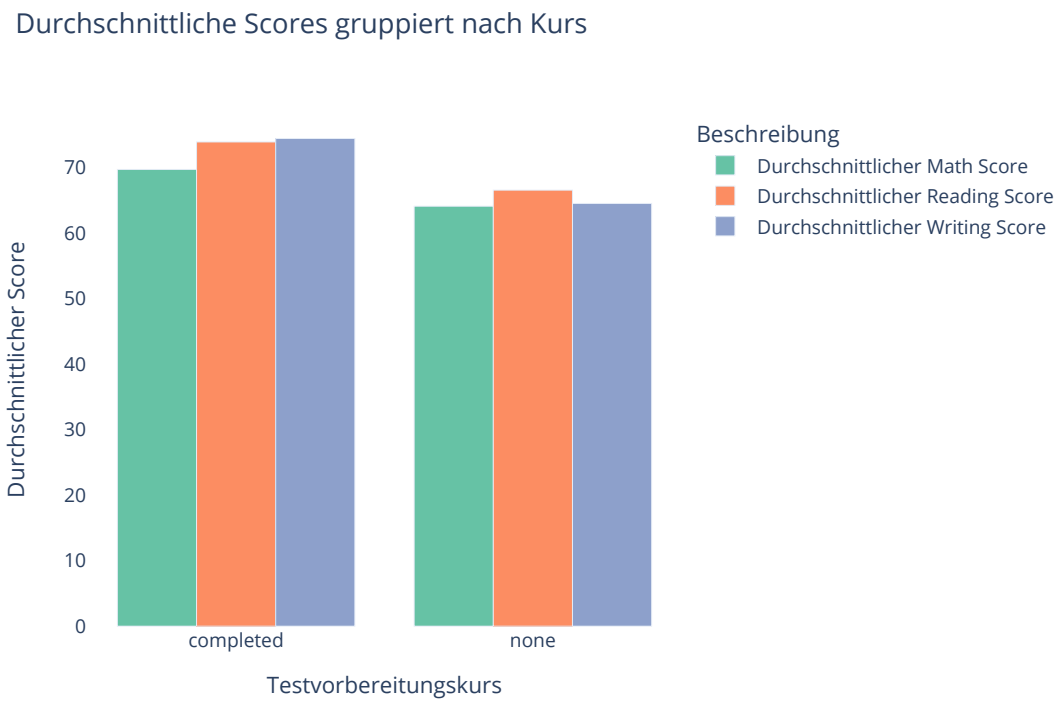
Wirkt sich die Absolvierung eines Vorbereitungskurses auf die Prüfungsergebnisse aus?

[14]:
```
average_scores = students_data.groupBy("test preparation course").agg(
    F.avg("math score").alias("Durchschnittlicher Math Score"),
    F.avg("reading score").alias("Durchschnittlicher Reading Score"),
    F.avg("writing score").alias("Durchschnittlicher Writing Score")
)

average_scores.show()
```

```
+----------------------+----------------------------+------------------------
-------+--------------------------------+
|test preparation course|Durchschnittlicher Math Score|Durchschnittlicher
Reading Score|Durchschnittlicher Writing Score|
+----------------------+----------------------------+------------------------
-------+--------------------------------+
|             completed|           69.69553072625699|
73.89385474860335|              74.41899441340782|
|                  none|            64.0778816199377|
66.53426791277259|              64.50467289719626|
+----------------------+----------------------------+------------------------
```

```
-------+--------------------------------+
```

[15]:
```
df = average_scores.toPandas()\
                   .sort_values('test preparation course')

fig = px.bar(df,
             x='test preparation course',
             y=['Durchschnittlicher Math Score', 'Durchschnittlicher Reading␣
 ↪Score', 'Durchschnittlicher Writing Score'],
             barmode='group',
             title='Durchschnittliche Scores gruppiert nach Kurs',
             labels={'test preparation course': 'Testvorbereitungskurs',␣
 ↪'variable': 'Beschreibung', 'value': 'Durchschnittlicher Score'},
             color_discrete_sequence=px.colors.qualitative.Set2)

fig.update_layout(paper_bgcolor="white", plot_bgcolor="white")
fig.show()
```



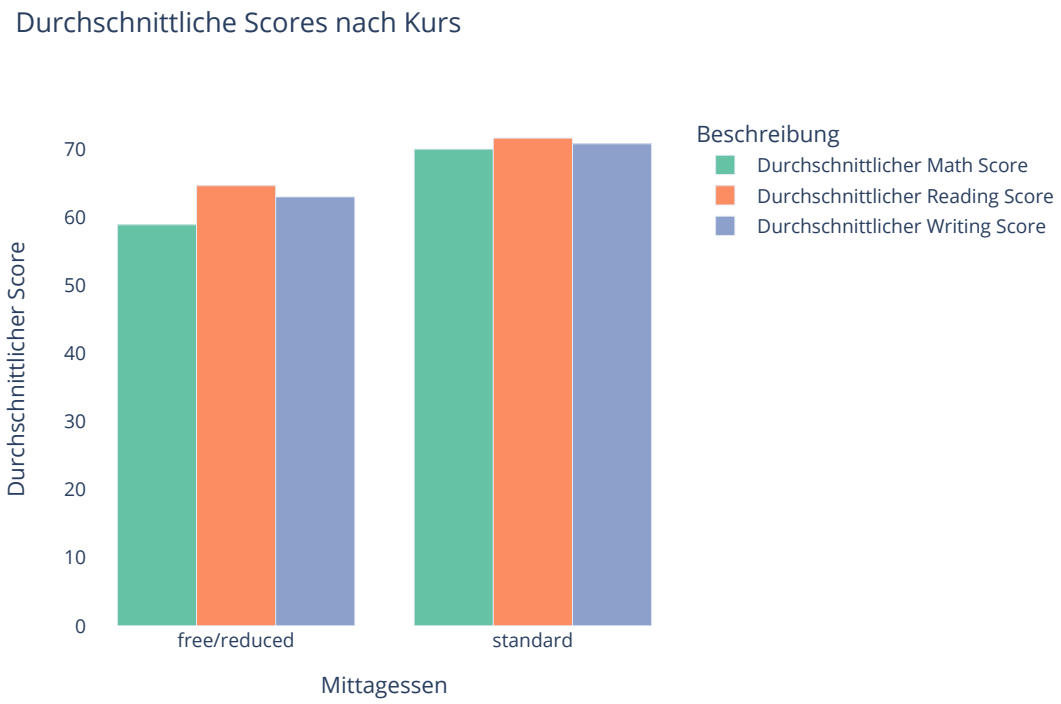
1.4 Forschungsfrage 3

Besteht ein statistisch signifikanter Zusammenhang zwischen der Bereitstellung eines Mittagessens und den Leistungen in Prüfungen, wobei höhere Prüfungsergebnisse bei Schülern beobachtet werden, die Zugang zu einem Mittagessen haben, im Vergleich zu Schülern ohne diesen Zugang?

[16]:
```
average_scores = students_data.groupBy("lunch").agg(
    F.avg("math score").alias("Durchschnittlicher Math Score"),
    F.avg("reading score").alias("Durchschnittlicher Reading Score"),
    F.avg("writing score").alias("Durchschnittlicher Writing Score")
)

average_scores.show()
```

```
+------------+-----------------------------+-------------------------------+---
-----------------------------+
|       lunch|Durchschnittlicher Math Score|Durchschnittlicher Reading
Score|Durchschnittlicher Writing Score|
+------------+-----------------------------+-------------------------------+---
-----------------------------+
|free/reduced|            58.92112676056338|              64.65352112676057|
63.02253521126761|
|    standard|            70.03410852713178|              71.65426356589147|
70.8232558139535|
+------------+-----------------------------+-------------------------------+---
-----------------------------+
```

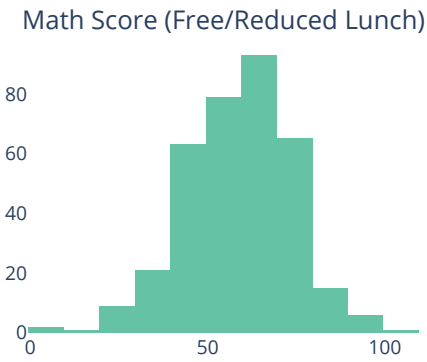
[17]:
```
df = average_scores.toPandas()\
                   .sort_values('lunch')

fig = px.bar(df,
             x='lunch',
             y=['Durchschnittlicher Math Score', 'Durchschnittlicher Reading␣
 ↪Score', 'Durchschnittlicher Writing Score'],
             barmode='group',
             title='Durchschnittliche Scores nach Kurs',
             labels={'lunch': 'Mittagessen', 'variable': 'Beschreibung',␣
 ↪'value': 'Durchschnittlicher Score'},
             color_discrete_sequence=px.colors.qualitative.Set2)

fig.update_layout(paper_bgcolor="white", plot_bgcolor="white")
fig.show()
```

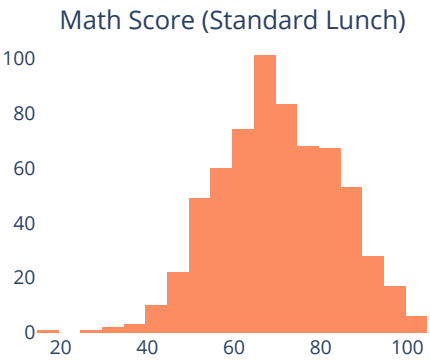


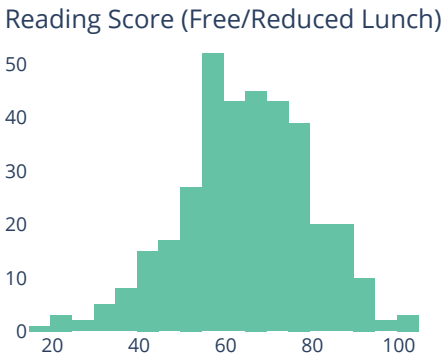
Test auf Normalverteilung um TTest durchführen zu können.

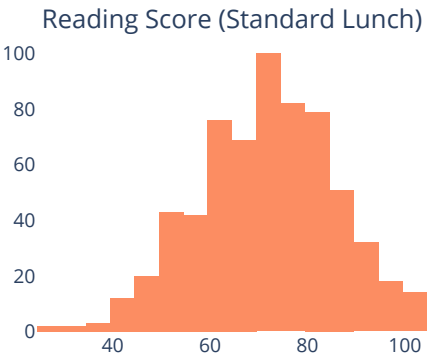
[18]:
```
free_reduced_math = students_data.filter(students_data["lunch"] == "free/
 ↪reduced").select("math score").rdd.flatMap(lambda x: x).collect()
free_reduced_reading = students_data.filter(students_data["lunch"] == "free/
 ↪reduced").select("reading score").rdd.flatMap(lambda x: x).collect()
free_reduced_writing = students_data.filter(students_data["lunch"] == "free/
 ↪reduced").select("writing score").rdd.flatMap(lambda x: x).collect()

standard_math = students_data.filter(students_data["lunch"] == "standard").
 ↪select("math score").rdd.flatMap(lambda x: x).collect()
standard_reading = students_data.filter(students_data["lunch"] == "standard").
 ↪select("reading Score").rdd.flatMap(lambda x: x).collect()
standard_writing = students_data.filter(students_data["lunch"] == "standard").
 ↪select("writing Score").rdd.flatMap(lambda x: x).collect()

mscore = 'Math Score'
rscore = 'Reading Score'
wscore = 'Writing Score'
free = 'Free/Reduced Lunch'
standard = 'Standard Lunch'
```

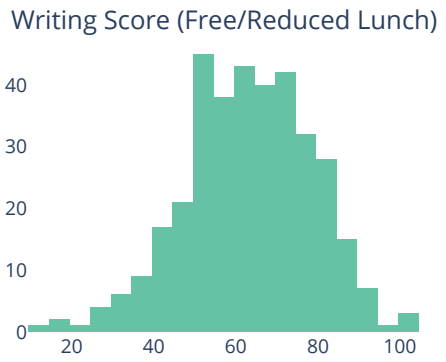
```
fig = make_subplots(rows=3, cols=2, subplot_titles=(f"{mscore} ({free})",␣
 ↪f"{mscore} ({standard})", f"{rscore} ({free})", f"{rscore} ({standard})",␣
 ↪f"{wscore} ({free})", f"{wscore} ({standard})"))

fig.add_trace(go.Histogram(x=free_reduced_math,
                           nbinsx=20,
                           name=f"{mscore} ({free})",
                           marker_color=px.colors.qualitative.Set2[0]),
                           row=1, col=1)
fig.add_trace(go.Histogram(x=standard_math,
                           nbinsx=20,
                           name=f"{mscore} ({standard})",
                           marker_color=px.colors.qualitative.Set2[1]),
                           row=1, col=2)
fig.add_trace(go.Histogram(x=free_reduced_reading,
                           nbinsx=20,
                           name=f"{rscore} ({free})",
                           marker_color=px.colors.qualitative.Set2[0]),
                           row=2, col=1)
fig.add_trace(go.Histogram(x=standard_reading,
                           nbinsx=20,
                           name=f"{rscore} ({standard})",
                           marker_color=px.colors.qualitative.Set2[1]),
                           row=2, col=2)
fig.add_trace(go.Histogram(x=free_reduced_writing,
                           nbinsx=20,
                           name=f"{wscore} ({free})",
                           marker_color=px.colors.qualitative.Set2[0]),
                           row=3, col=1)
fig.add_trace(go.Histogram(x=standard_writing,
                           nbinsx=20,
                           name=f"{wscore} ({standard})",
                           marker_color=px.colors.qualitative.Set2[1]),
                           row=3, col=2)

fig.update_layout(showlegend=False, height=1000, paper_bgcolor="white",␣
 ↪plot_bgcolor="white")
fig.show()
```

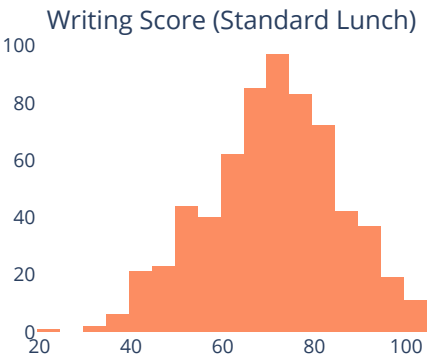

Math Score (Free/Reduced Lunch)
80
60
40
20
0
0
50
100


Math Score (Standard Lunch)
100
80
60
40
20
0
20
40
60
80
100


Reading Score (Free/Reduced Lunch)
50
40
30
20
10
0
20
40
60
80
100


Reading Score (Standard Lunch)
100
80
60
40
20
0
40
60
80
100


Writing Score (Free/Reduced Lunch)
40
30
20
10
0
20
40
60
80
100


Writing Score (Standard Lunch)
100
80
60
40
20
0
20
40
60
80
100

Vorbereitung der Daten

[19]:
```
standard_lunch_math_scores = students_data.filter(F.col("lunch") == "standard").
 ↪select("math score").rdd.flatMap(lambda x: x).collect()
no_lunch_math_scores = students_data.filter(F.col("lunch") == "free/reduced").
 ↪select("math score").rdd.flatMap(lambda x: x).collect()
standard_lunch_reading_scores = students_data.filter(F.col("lunch") ==␣
 ↪"standard").select("reading score").rdd.flatMap(lambda x: x).collect()
no_lunch_reading_scores = students_data.filter(F.col("lunch") == "free/
 ↪reduced").select("reading score").rdd.flatMap(lambda x: x).collect()
standard_lunch_writing_scores = students_data.filter(F.col("lunch") ==␣
 ↪"standard").select("writing score").rdd.flatMap(lambda x: x).collect()
no_lunch_writing_scores = students_data.filter(F.col("lunch") == "free/
 ↪reduced").select("writing score").rdd.flatMap(lambda x: x).collect()
```

[20]:
```
t_test_math = stats.ttest_ind(standard_lunch_math_scores, no_lunch_math_scores)
t_test_reading = stats.ttest_ind(standard_lunch_reading_scores,␣
 ↪no_lunch_reading_scores)
t_test_writing = stats.ttest_ind(standard_lunch_writing_scores,␣
 ↪no_lunch_writing_scores)

print(f"{'Math Scores T-Test:':<22}{t_test_math[0]:.2f} (p-value:␣
 ↪{t_test_math[1]:.2f})")
print(f"{'Reading Scores T-Test:':<22} {t_test_reading[0]:.2f} (p-value:␣
 ↪{t_test_reading[1]:.2f})")
print(f"{'Writing Scores T-Test:':<22} {t_test_writing[0]:.2f} (p-value:␣
 ↪{t_test_writing[1]:.2f})")
```

```
Math Scores T-Test:   11.84 (p-value: 0.00)
Reading Scores T-Test: 7.45 (p-value: 0.00)
Writing Scores T-Test: 8.01 (p-value: 0.00)
```

1.5 Forschungsfrage 4 - Machine Learning

1.5.1 Kann unter Verwendung der Variablen 'Parental Level of Education', 'Lunch' und 'Test Preperation Course' mittels einer ausgewählten Machine Learning Methode die Note des Prüfungsergebnisses vorhergesagt werden, wobei die Prüfungsergebnisse in Noten nach dem österreichischen Schulsystem (1-5) eingeteilt werden?

[21]:
```
# Let's start again with a little overview of the dataset
students_data.show()
```

```
+------+--------------+--------------------------+------------+----------------
-------+----------+-------------+-------------+
|gender|race/ethnicity|parental level of education|       lunch|test preparation
course|math score|reading score|writing score|
+------+--------------+--------------------------+------------+----------------
-------+----------+-------------+-------------+
```

```
|female|       group B|          bachelor's degree|    standard|
none|        72|           72|           74|
|female|       group C|               some college|    standard|
completed|        69|           90|           88|
|female|       group B|             master's degree|    standard|
none|        90|           95|           93|
|  male|       group A|          associate's degree|free/reduced|
none|        47|           57|           44|
|  male|       group C|               some college|    standard|
none|        76|           78|           75|
|female|       group B|          associate's degree|    standard|
none|        71|           83|           78|
|female|       group B|               some college|    standard|
completed|        88|           95|           92|
|  male|       group B|               some college|free/reduced|
none|        40|           43|           39|
|  male|       group D|                high school|free/reduced|
completed|        64|           64|           67|
|female|       group B|                high school|free/reduced|
none|        38|           60|           50|
|  male|       group C|          associate's degree|    standard|
none|        58|           54|           52|
|  male|       group D|          associate's degree|    standard|
none|        40|           52|           43|
|female|       group B|                high school|    standard|
none|        65|           81|           73|
|  male|       group A|               some college|    standard|
completed|        78|           72|           70|
|female|       group A|             master's degree|    standard|
none|        50|           53|           58|
|female|       group C|            some high school|    standard|
none|        69|           75|           78|
|  male|       group C|                high school|    standard|
none|        88|           89|           86|
|female|       group B|            some high school|free/reduced|
none|        18|           32|           28|
|  male|       group C|             master's degree|free/reduced|
completed|        46|           42|           46|
|female|       group C|          associate's degree|free/reduced|
none|        54|           58|           61|
+------+--------------+---------------------------+------------+----------------
-------+----------+-------------+-------------+
only showing top 20 rows
```

[22]:
```
categorical_columns = ['gender', 'race/ethnicity', 'parental level of␣
 ↪education', 'lunch', 'test preparation course']
```

```
stages = []
```

[23]:
```
for column in categorical_columns:
    stringIndexer = StringIndexer(inputCol=column, outputCol=column + 'Index')
    encoder = OneHotEncoder(inputCols=[stringIndexer.getOutputCol()],␣
 ↪outputCols=[column + "classVec"])
    stages += [stringIndexer, encoder]
```

Learning: Pyspark requires a single feature vector

Learning: Pyspark unterstützt keine multi-output regression probleme.

[24]:
```
label_column = 'math score'
assemblerInputs = [c + "classVec" for c in categorical_columns] + ['reading␣
 ↪score', 'writing score'] # Definition der Inputs für den Assembler
assembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features") #␣
 ↪Vektor für den Input in ML-Model ist features
stages += [assembler]
```

[25]:
```
pipeline = Pipeline(stages=stages)
pipelineModel = pipeline.fit(students_data) # Anwendung der Estimator Schritte␣
 ↪werden in der Pipeline auf die Daten trainiert
students_data_transf = pipelineModel.transform(students_data)
```

[26]:
```
train, test = students_data_transf.randomSplit([0.7, 0.3], seed=2021)
```

[27]:
```
lr = LinearRegression(featuresCol='features', labelCol=label_column)
lrModel = lr.fit(train)
```

```
24/01/22 10:22:57 WARN Instrumentation: [af450525] regParam is zero, which might
cause numerical instability and overfitting.
24/01/22 10:22:57 WARN InstanceBuilder: Failed to load implementation
from:dev.ludovic.netlib.blas.JNIBLAS
24/01/22 10:22:57 WARN InstanceBuilder: Failed to load implementation
from:dev.ludovic.netlib.blas.VectorBLAS
24/01/22 10:22:57 WARN InstanceBuilder: Failed to load implementation
from:dev.ludovic.netlib.lapack.JNILAPACK
```

Evaluierung Linear Regression Model

[28]:
```
predictions = lrModel.transform(test)
evaluator = RegressionEvaluator(labelCol=label_column,␣
 ↪predictionCol="prediction", metricName = "rmse")

rmse = evaluator.evaluate(predictions)
print(f"Root Mean Squared Error (RMSE) on test data = {rmse}")
```

```
Root Mean Squared Error (RMSE) on test data = 5.459658482750319
```

Einteilung österreichisches Schulsystem

- 1: 100 - 90
- 2: 89 - 80
- 3: 79 - 65
- 4: 64 - 50
- 5: 49 - 0

[29]:
```
def score_to_grade(score):
    if 90 <= score <= 100:
        return 1
    elif 80 <= score <= 89:
        return 2
    elif 65 <= score <= 79:
        return 3
    elif 50 <= score <= 64:
        return 4
    elif 0 <= score <= 49:
        return 5
    else:
        return None

score_to_grade_udf = F.udf(score_to_grade, IntegerType()) # UDF müssen wir für␣
 ↪die Anwendung auf die Spalten im Dataframe registrieren
```

[30]:
```
students_data = students_data.withColumn("math_grade", score_to_grade_udf("math␣
 ↪score"))
students_data = students_data.withColumn("reading_grade",␣
 ↪score_to_grade_udf("reading score"))
students_data = students_data.withColumn("writing_grade",␣
 ↪score_to_grade_udf("writing score"))

students_data.show()
```

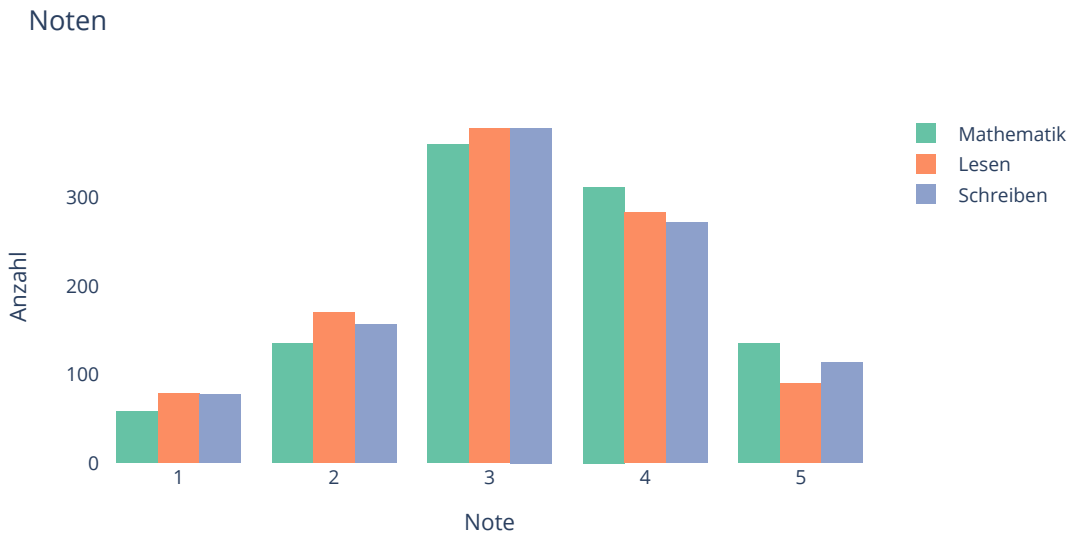
```
+------+--------------+--------------------------+------------+----------------
-------+----------+-------------+-------------+----------+-------------+--------
-----+
|gender|race/ethnicity|parental level of education|       lunch|test preparation
course|math score|reading score|writing
score|math_grade|reading_grade|writing_grade|
+------+--------------+--------------------------+------------+----------------
-------+----------+-------------+-------------+----------+-------------+--------
-----+
|female|       group B|          bachelor's degree|    standard|
none|        72|           72|           74|         3|            3|
3|
|female|       group C|               some college|    standard|
completed|        69|           90|           88|         3|            1|
```

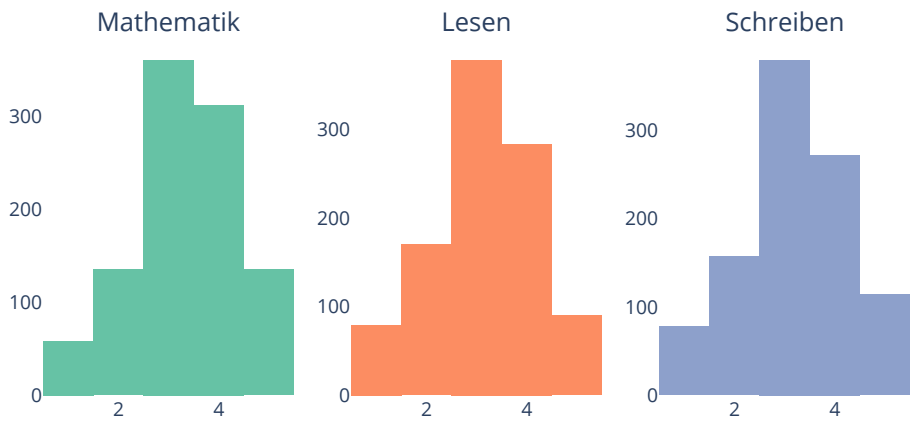
```
2|
|female|       group B|           master's degree|    standard|
none|        90|           95|           93|         1|            1|
1|
|  male|       group A|         associate's degree|free/reduced|
none|        47|           57|           44|         5|            4|
5|
|  male|       group C|              some college|    standard|
none|        76|           78|           75|         3|            3|
3|
|female|       group B|         associate's degree|    standard|
none|        71|           83|           78|         3|            2|
3|
|female|       group B|              some college|    standard|
completed|        88|           95|           92|         2|            1|
1|
|  male|       group B|              some college|free/reduced|
none|        40|           43|           39|         5|            5|
5|
|  male|       group D|               high school|free/reduced|
completed|        64|           64|           67|         4|            4|
3|
|female|       group B|               high school|free/reduced|
none|        38|           60|           50|         5|            4|
4|
|  male|       group C|         associate's degree|    standard|
none|        58|           54|           52|         4|            4|
4|
|  male|       group D|         associate's degree|    standard|
none|        40|           52|           43|         5|            4|
5|
|female|       group B|               high school|    standard|
none|        65|           81|           73|         3|            2|
3|
|  male|       group A|              some college|    standard|
completed|        78|           72|           70|         3|            3|
3|
|female|       group A|           master's degree|    standard|
none|        50|           53|           58|         4|            4|
4|
|female|       group C|         some high school|    standard|
none|        69|           75|           78|         3|            3|
3|
|  male|       group C|               high school|    standard|
none|        88|           89|           86|         2|            2|
2|
|female|       group B|         some high school|free/reduced|
none|        18|           32|           28|         5|            5|
```

```
5|
|  male|       group C|            master's degree|free/reduced|
completed|        46|           42|           46|         5|            5|
5|
|female|       group C|         associate's degree|free/reduced|
none|        54|           58|           61|         4|            4|
4|
+------+-------------+--------------------------+------------+---------------
-------+----------+-------------+-------------+----------+-------------+--------
-----+
only showing top 20 rows
```

[31]:
```
# Visualisierung
math_grades = students_data.select("math_grade").rdd.flatMap(lambda x: x).
 ↪collect()
reading_grades = students_data.select("reading_grade").rdd.flatMap(lambda x: x).
 ↪collect()
writing_grades = students_data.select("writing_grade").rdd.flatMap(lambda x: x).
 ↪collect()

trace1 = go.Histogram(x=math_grades,
                      name="Mathematik",
                      xbins=dict(
                              start=0.5,
                              end=5.5),
                      marker_color=px.colors.qualitative.Set2[0])
trace2 = go.Histogram(x=reading_grades,
                      name="Lesen",
                      xbins=dict(
                              start=0.5,
                              end=5.5),
                      marker_color=px.colors.qualitative.Set2[1])
trace3 = go.Histogram(x=writing_grades,
                      name="Schreiben",
                      xbins=dict(
                          start=0.5,
                          end=5.5),
                      marker_color=px.colors.qualitative.Set2[2])

data = [trace1, trace2, trace3]

fig = go.Figure(data=data, layout=go.Layout(barmode='group', title='Noten',␣
 ↪xaxis_title='Note', yaxis_title='Anzahl'))

fig.update_layout(showlegend=True, height=400, paper_bgcolor="white",␣
 ↪plot_bgcolor="white")
```

```
fig.show()

fig = make_subplots(rows=1, cols=3, subplot_titles=("Mathematik", "Lesen",␣
 ↪"Schreiben"))

fig.add_trace(trace1, row=1, col=1)
fig.add_trace(trace2, row=1, col=2)
fig.add_trace(trace3, row=1, col=3)

fig.update_layout(showlegend=False, height=400, paper_bgcolor="white",␣
 ↪plot_bgcolor="white")
fig.show()
```





[32]:
```
mean_math_grade = students_data.agg(F.mean("math_grade")).collect()[0][0]
print(f"{'Durchschnitt von math_grade:':<31} {mean_math_grade:.2f}")

mean_reading_grade = students_data.agg(F.mean("reading_grade")).collect()[0][0]
print(f"{'Durchschnitt von reading_grade:':<31} {mean_reading_grade:.2f}")

mean_writing_grade = students_data.agg(F.mean("writing_grade")).collect()[0][0]
print(f"{'Durchschnitt von writing_grade:':<31} {mean_writing_grade:.2f}")
```

```
Durchschnitt von math_grade:    3.33
Durchschnitt von reading_grade: 3.13
Durchschnitt von writing_grade: 3.19
```

1.5.2 Predicten der Schulnoten

[33]:
```
categorical_columns = ['parental level of education', 'lunch', 'test␣
 ↪preparation course']
stages_math = []
stages_reading = []
stages_writing = []
```

[34]:
```
# Learning: Probleme wenn man den selben dataframe in mehreren Pipelines␣
 ↪verwendet

students_data_1 = students_data
students_data_2 = students_data
```

```
students_data_3 = students_data
```

Learning Math Grade

[35]:
```
for column in categorical_columns:
    stringIndexer = StringIndexer(inputCol=column, outputCol=column + 'Index')
    encoder = OneHotEncoder(inputCols=[stringIndexer.getOutputCol()],␣
 ↪outputCols=[column + "classVec"])
    stages_math += [stringIndexer, encoder]
```

[36]:
```
label_column = 'math_grade'
assemblerInputs = [c + "classVec" for c in categorical_columns] # Definition␣
 ↪der Inputs für den Assembler
assembler = VectorAssembler(inputCols=assemblerInputs,␣
 ↪outputCol="features_math") # Vektor für den Input in ML-Model ist features
stages_math += [assembler]
```

[37]:
```
pipeline = Pipeline(stages=stages_math)
pipelineModel = pipeline.fit(students_data_1) # Anwendung der Estimator␣
 ↪Schritte werden in der Pipeline auf die Daten trainiert
students_data_math = pipelineModel.transform(students_data_1)

# Fehlermeldung: IllegalArgumentException: OutputColumnFeatures already exists
# Haben deshalb stages_math, stages_reading, stages_writing erstellt.
```

[38]:
```
train, test = students_data_math.randomSplit([0.7, 0.3], seed=2021)

lr = LinearRegression(featuresCol='features_math', labelCol=label_column)
lrModel = lr.fit(train)
```

```
24/01/22 10:23:00 WARN Instrumentation: [1c63d67e] regParam is zero, which might
cause numerical instability and overfitting.
```

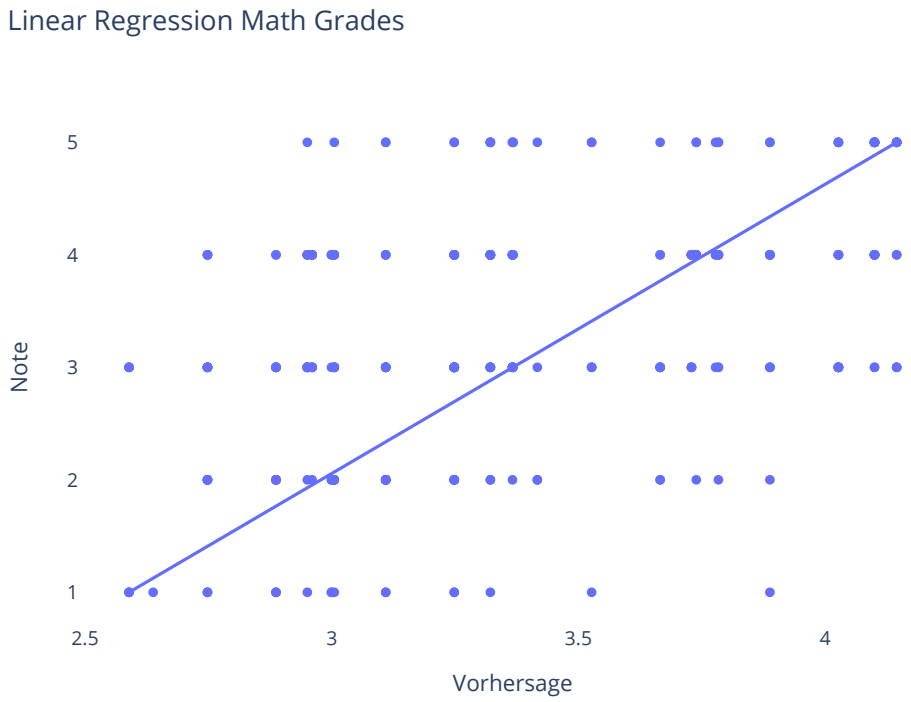
Evaluierung Math

[39]:
```
predictions = lrModel.transform(test)
evaluator = RegressionEvaluator(labelCol=label_column,␣
 ↪predictionCol="prediction", metricName = "rmse")

rmse = evaluator.evaluate(predictions)
print(f"Root Mean Squared Error (RMSE) on test data = {rmse}")
```

```
Root Mean Squared Error (RMSE) on test data = 0.9858671685131497
```

[40]:
```
predictions_pd = predictions.toPandas()
fig = px.scatter(predictions_pd, x="prediction", y="math_grade", title="Linear␣
 ↪Regression Math Grades", labels={'prediction': 'Vorhersage', 'math_grade':␣
 ↪'Note'})
```

```
fig.add_trace(px.line(x=[predictions_pd["prediction"].min(),␣
 ↪predictions_pd["prediction"].max()], y=[predictions_pd[label_column].min(),␣
 ↪predictions_pd[label_column].max()], line_shape="linear").data[0])
fig.update_layout(paper_bgcolor="white", plot_bgcolor="white")
fig.show()
```



Learning Reading Grade

[41]:
```
for column in categorical_columns:
    stringIndexer = StringIndexer(inputCol=column, outputCol=column + 'Index')
    encoder = OneHotEncoder(inputCols=[stringIndexer.getOutputCol()],␣
 ↪outputCols=[column + "classVec"])
    stages_reading += [stringIndexer, encoder]
```

[42]:
```
label_column = 'reading_grade'
assemblerInputs = [c + "classVec" for c in categorical_columns] # Definition␣
 ↪der Inputs für den Assembler
assembler = VectorAssembler(inputCols=assemblerInputs,␣
 ↪outputCol="features_reading") # Vektor für den Input in ML-Model ist features
stages_reading += [assembler]
```

[43]:
```
pipeline = Pipeline(stages=stages_reading)
pipelineModel = pipeline.fit(students_data_2) # Anwendung der Estimator␣
 ↪Schritte werden in der Pipeline auf die Daten trainiert
students_data_reading = pipelineModel.transform(students_data_2)
```

[44]:
```
train, test = students_data_reading.randomSplit([0.7, 0.3], seed=2021)

lr = LinearRegression(featuresCol='features_reading', labelCol=label_column)
lrModel = lr.fit(train)
```

```
24/01/22 10:23:02 WARN Instrumentation: [fa13f2ae] regParam is zero, which might
cause numerical instability and overfitting.
```

Evaluierung Reading

[45]:
```
predictions = lrModel.transform(test)
evaluator = RegressionEvaluator(labelCol=label_column,␣
 ↪predictionCol="prediction", metricName = "rmse")

rmse = evaluator.evaluate(predictions)
print(f"Root Mean Squared Error (RMSE) on test data = {rmse}")
```

```
Root Mean Squared Error (RMSE) on test data = 0.9807520340124966
```

[46]:
```
predictions_pd = predictions.toPandas()
fig = px.scatter(predictions_pd, x="prediction", y="reading_grade",␣
 ↪title="Linear Regression Reading Grades", labels={'prediction':␣
 ↪'Vorhersage', 'reading_grade': 'Note'})
fig.add_trace(px.line(x=[predictions_pd["prediction"].min(),␣
 ↪predictions_pd["prediction"].max()], y=[predictions_pd[label_column].min(),␣
 ↪predictions_pd[label_column].max()], line_shape="linear").data[0])
fig.update_layout(paper_bgcolor="white", plot_bgcolor="white")
fig.show()
```

Learning Writing Grade

[47]:
```
for column in categorical_columns:
    stringIndexer = StringIndexer(inputCol=column, outputCol=column + 'Index')
    encoder = OneHotEncoder(inputCols=[stringIndexer.getOutputCol()],␣
 ↪outputCols=[column + "classVec"])
    stages_writing += [stringIndexer, encoder]
```

[48]:
```
label_column = 'writing_grade'
assemblerInputs = [c + "classVec" for c in categorical_columns] # Definition␣
 ↪der Inputs für den Assembler
assembler = VectorAssembler(inputCols=assemblerInputs,␣
 ↪outputCol="features_writing") # Vektor für den Input in ML-Model ist features
stages_writing += [assembler]
```

[49]:
```
pipeline = Pipeline(stages=stages_writing)
pipelineModel = pipeline.fit(students_data_3) # Anwendung der Estimator␣
 ↪Schritte werden in der Pipeline auf die Daten trainiert
students_data_writing = pipelineModel.transform(students_data_3)
```

[50]:
```
train, test = students_data_writing.randomSplit([0.7, 0.3], seed=2021)
```

```
lr = LinearRegression(featuresCol='features_writing', labelCol=label_column)
lrModel = lr.fit(train)
```

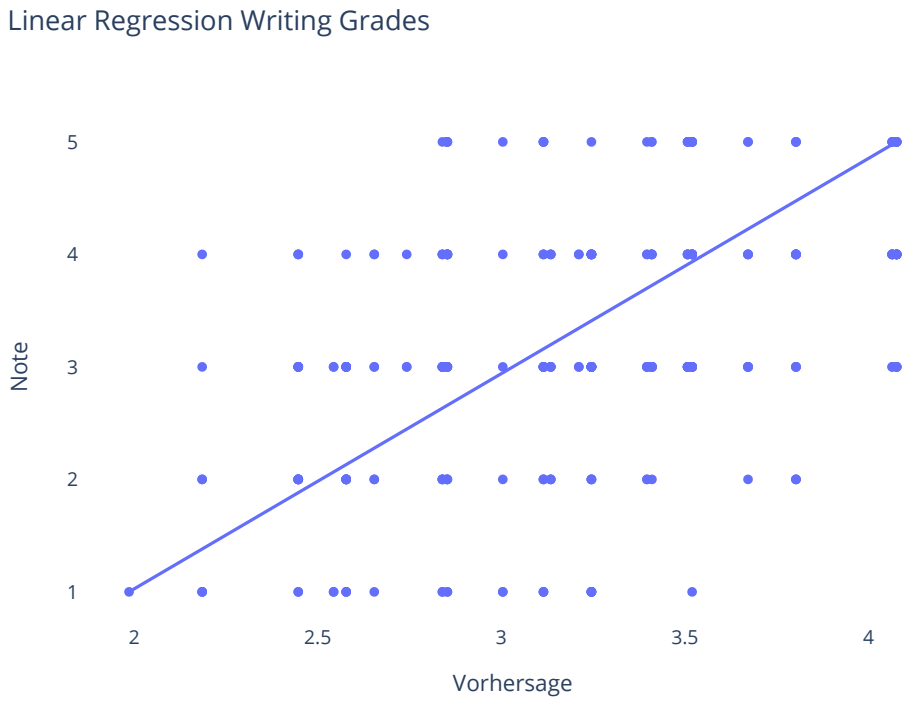
```
24/01/22 10:23:03 WARN Instrumentation: [9472f194] regParam is zero, which might
cause numerical instability and overfitting.
```

Evaluierung Writing

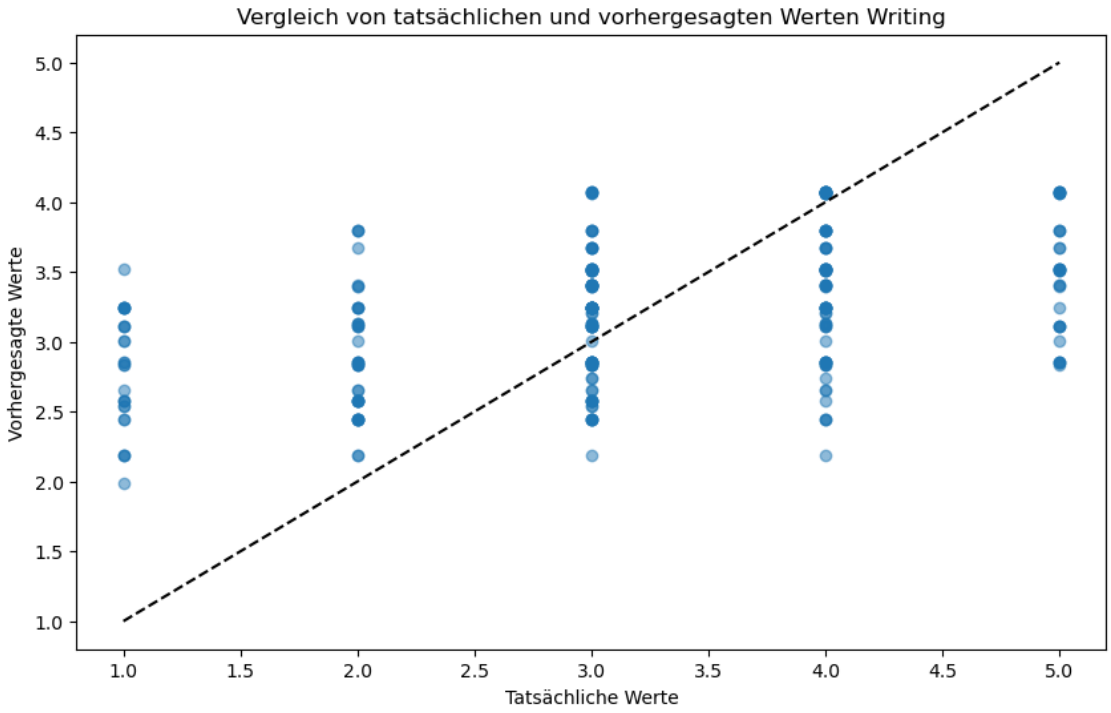
[51]:
```
predictions = lrModel.transform(test)
evaluator = RegressionEvaluator(labelCol=label_column,␣
 ↪predictionCol="prediction", metricName = "rmse")

rmse = evaluator.evaluate(predictions)
print(f"Root Mean Squared Error (RMSE) on test data = {rmse}")
```

```
Root Mean Squared Error (RMSE) on test data = 0.9999110082573022
```

[52]:
```
predictions_pd = predictions.toPandas()
fig = px.scatter(predictions_pd, x="prediction", y="writing_grade",␣
 ↪title="Linear Regression Writing Grades", labels={'prediction':␣
 ↪'Vorhersage', 'writing_grade': 'Note'})
fig.add_trace(px.line(x=[predictions_pd["prediction"].min(),␣
 ↪predictions_pd["prediction"].max()], y=[predictions_pd[label_column].min(),␣
 ↪predictions_pd[label_column].max()], line_shape="linear").data[0])
fig.update_layout(paper_bgcolor="white", plot_bgcolor="white")
fig.show()
```



[53]:
```
plt.figure(figsize=(10, 6))
plt.scatter(predictions_pd[label_column], predictions_pd["prediction"], alpha=0.
 ↪5)
plt.xlabel('Tatsächliche Werte')
plt.ylabel('Vorhergesagte Werte')
plt.title('Vergleich von tatsächlichen und vorhergesagten Werten Writing')
plt.plot([predictions_pd[label_column].min(), predictions_pd[label_column].
 ↪max()], [predictions_pd[label_column].min(), predictions_pd[label_column].
 ↪max()], 'k--')
plt.show()
```



2 Fazit

Unser LabProject repräsentiert eine umfassende Untersuchung von Schülerleistungen unter Verwendung eines Datensatzes von Kaggle. Ziel war es, mit Big Data Werkzeugen wie Apache Spark und Visualisierungstools wie Plotly wertvolle Erkenntnisse zu gewinnen und diese effektiv zu kommunizieren.

Die Analyse konzentrierte sich auf vier Hauptforschungsfragen, die Unterschiede in den Prüfungsergebnissen zwischen verschiedenen ethnischen Gruppen, den Einfluss des Bildungsniveaus der Eltern auf die Absolvierung von Vorbereitungskursen und deren Auswirkungen auf die Prü-

fungsergebnisse, den Zusammenhang zwischen Mittagessen und Leistung und die Vorhersage von Noten mit maschinellem Lernen umfassten.

Die Ergebnisse zeigten deutliche Leistungsunterschiede zwischen den ethnischen Gruppen, wobei Gruppe E die besten und Gruppe A die niedrigsten Durchschnittswerte aufwies. Eltern mit höherem Bildungsniveau schicken ihre Kinder nicht häufiger zu Vorbereitungskursen als Eltern mit niedrigerem Bildungsniveas. Die Absolvierung eines Vorbereitungskurses beeinflusst die Prüfungsergebnisse positiv. Darüber hinaus wurde festgestellt, dass Schüler mit Standardmittagessen durchschnittlich besser abschnitten als ihre Kollegen mit reduziertem Mittagessen.

Mittels Machine Learning Modellen konnten die Noten basierend auf den Variablen 'Parental Level of Education', 'Lunch' und 'Test Preparation Course' mit einer guten Genauigkeit vorhergesagt werden, wie durch die RMSE-Werte belegt wird. Die Visualisierung der Ergebnisse trug wesentlich zum Verständnis der analysierten Trends bei.

Insgesamt bietet das Projekt tiefgreifende Einblicke in die schulische Leistung und verdeutlicht die Bedeutung einer ausgewogenen Ernährung. Die Studie unterstreicht auch die Relevanz von Vorbereitungskursen für die akademische Leistung der Schüler.