Data Science 874
Post Block Assignment 3

Jeandre de Bruyn
19768206

Business Understanding

This dataset is taken from Kaggle at the following URL:
https://www.kaggle.com/spscientist/students-performance-in-exams

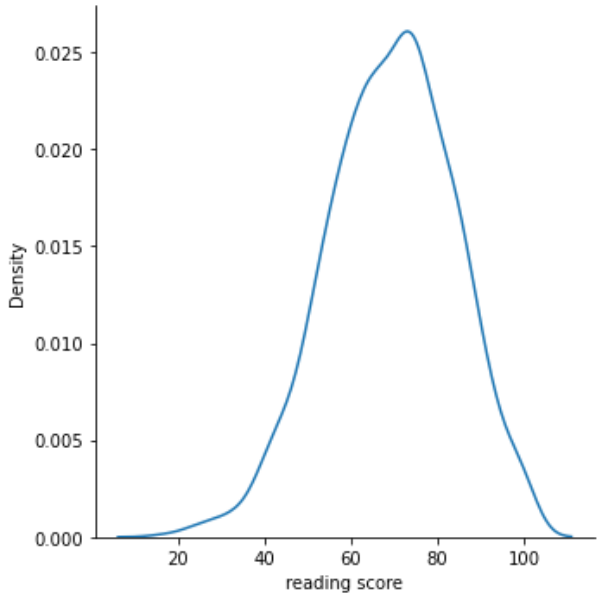
The relationship between student background and their results are of great interest to the education departments of the world because the sooner that you can predict or anticipate a learner that may have difficulty at school the sooner that action can be taken support given to the learner to ensure that they can fulfil their academic potential.

As a result, the main objective of this modelling exercise will be to predict whether a student needs academic assistance by predicting whether they will score above or below average given the background factors such as their parents level of education and whether the families financial status qualifies them for free/reduced school lunches.

Feature Exploration

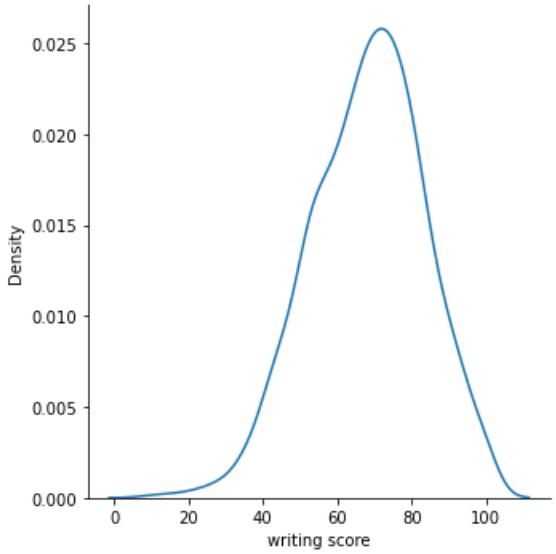
There are three continuous features, each one describing the writing, reading and math score of a student. Five categorical features describe the socioeconomic background of the student such as gender, parental education etc. In this dataset, the race feature has been renamed to groups A-D. The true meaning of these groups is unknown.

Reading score



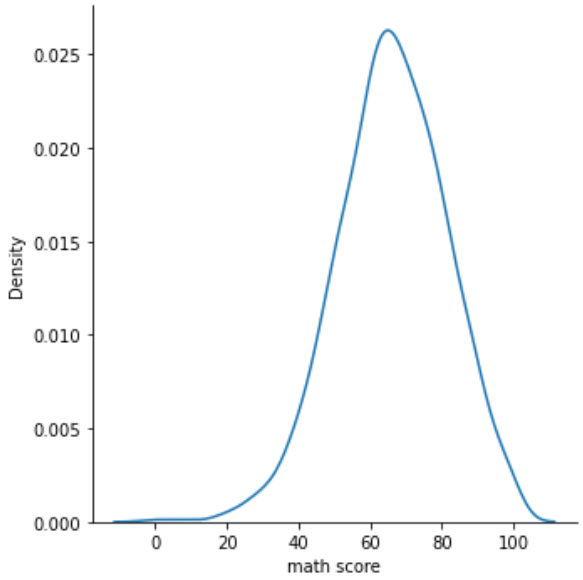
We see a normal distribution centred around 70 with a slight tail to the left

Writing score



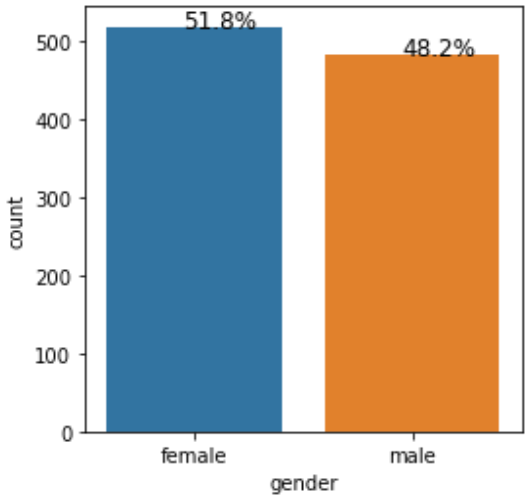
Similar to the reading score there is a normal distribution centred around 70 with a slight tail to the left.

Math score



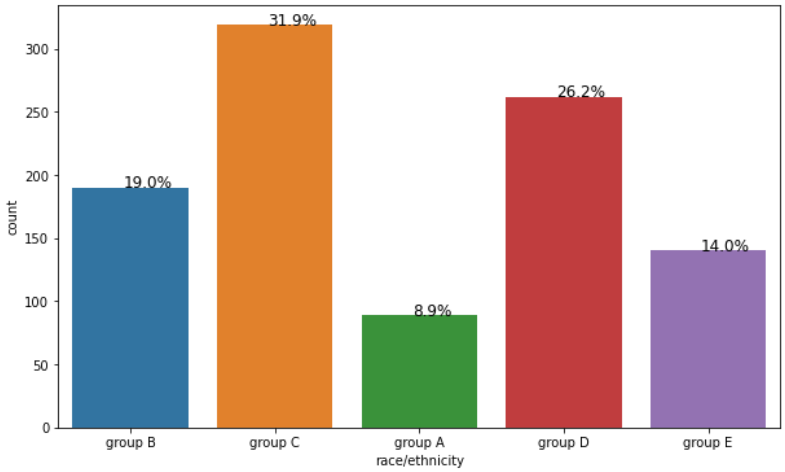
Math score has a normal distribution centred in the low 60s with a slight tail to the left.

Gender



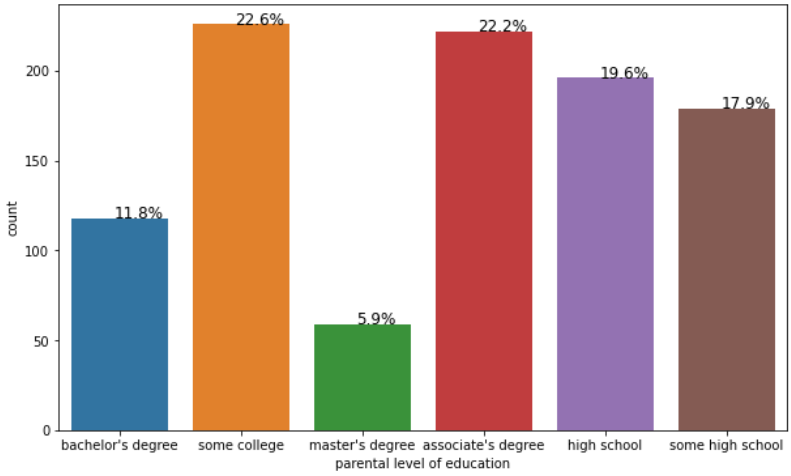
The gender distribution is essentially equal with a slight majority of females in the dataset

Race



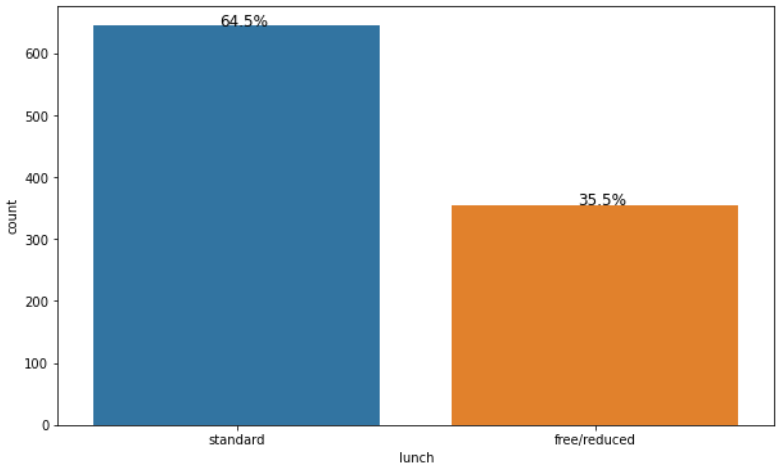
It is unknown what the groups stand for but there is a clear majority where two groups (C and D) hold over 50% of the dataset.

Parental level of education



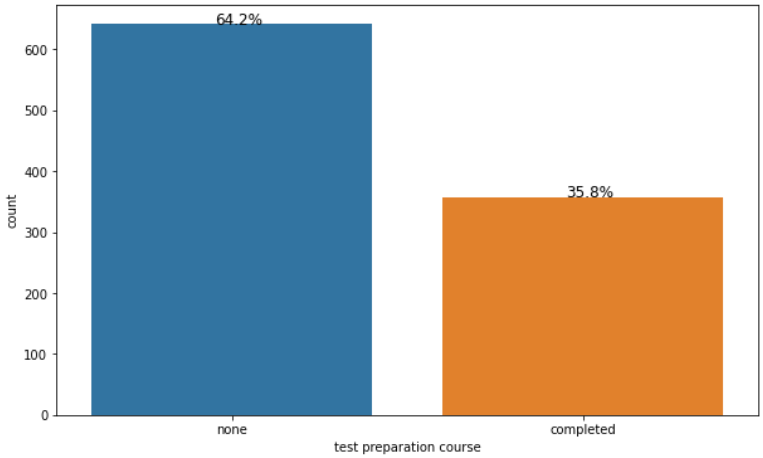
Nearly 40% of all students' parents only have some level of high school, whether that be partial or full.

Lunch



Just over 1/3rd of all students qualifies for free or reduced school lunch costs. The way that students normally qualify for this is based on parental income.

Test Preparation course



A similar proportion, when compared to school lunches, has not completed any kind of test preparation courses, just over 1/3rd.

Data Quality report:

Feature	Count	Miss %	Card.	Min.	1st Qrt.	Mean	Median	3rd Qrt.	Max	Std. Dev.
Math score	1000	0	81	0	57	66.08	66	77	100	15.16
reading score	1000	0	72	17	59	69	70	79	100	14.6
writing score	1000	0	77	10	57.75	68.05	69	79	100	15.2

Table 1: Continuous feature data quality report

Below is an analysis of the distances between the min, max and quartiles

	min-1st	1st-med	med-3	3-max
math score	57	9	11	23
reading score	42	11	9	21
writing score	47.75	11.25	10	21

There is a large gap as expected between the minimum and the 1st quartile due to the tail that can be seen to the left of each score's distribution.

Feature	Count	Miss %	Card.	Mode	Mode Freq.	Mode%	2nd Mode	2nd Mode Freq.	2nd Mode %
gender	1000	0	2	female	518	51.80%	male	482	48.20%
race	1000	0	5	group C	319	31.90%	group d	262	26.20%
parental level of education	1000	0	6	some college	226	22.60%	associates degree	222	22.20%
lunch	1000	0	2	standard	645	64.50%	free/reduced	335	33.50%
test prep course	1000	0	2	none	652	65.20%	completed	358	35.80%

Table 2: Categorical feature data quality report

Feature	Data quality issue	Potential Handling Strategies
math score	Outliers (low)	Keep outliers since they are valid
reading score	Outliers (low)	Keep outliers since they are valid
writing score	Outliers (low)	Keep outliers since they are valid

Table 3: Table showing data quality issues and their handling strategies

Data Preparation

The goal of this exercise is to use the socioeconomic background of students to predict whether or not a student is in need of academic assistance, however, it is unnecessary to have 3 target features (the writing, reading and math scores), instead, a new feature will be created which is the aggregate of these three scores. This new "average" variable will be the target variable for the modelling process.

The categorical variables will be made more usable in the modelling stage by using dummy coding. At the same time, the average score of the learners is split into two separate categories, being if they are below or above the median and average of 68. This makes the predictions more accurate than trying to predict an exact score while also not detracting any usefulness from the prediction.

Visualization

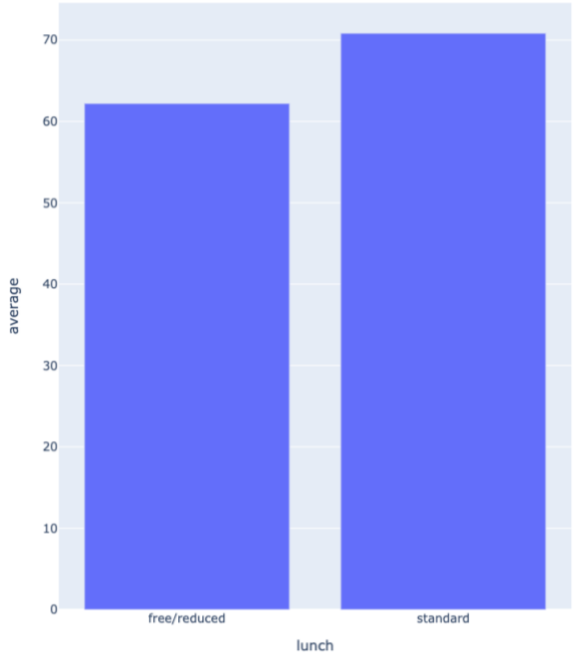


Figure 1: Bar plot comparing average score between students that have standard lunch rates and not

The above figure shows that students on a free/reduced lunch plan score lower on average than those students that fall into the standard lunch category, there is a difference of just under 10 average scores between the two categories, the difference in distribution can be seen in more detail in the box plot comparing the two below.

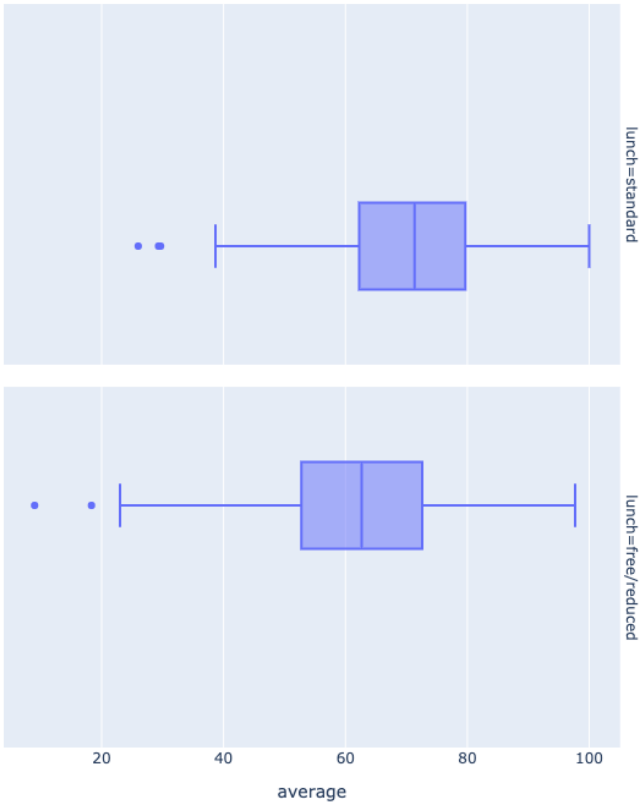


Figure 2: Box plot comparing students under standard lunch plan and not

From the box plot, we can see that the free/reduced distribution has a lower Q1, Median and Q3 than the standard lunch distribution showing that there is a substantial difference between the two populations that when combined with other features can lead to high predictive power.

Below is a boxplot exploring the different ethnicities in the dataset.

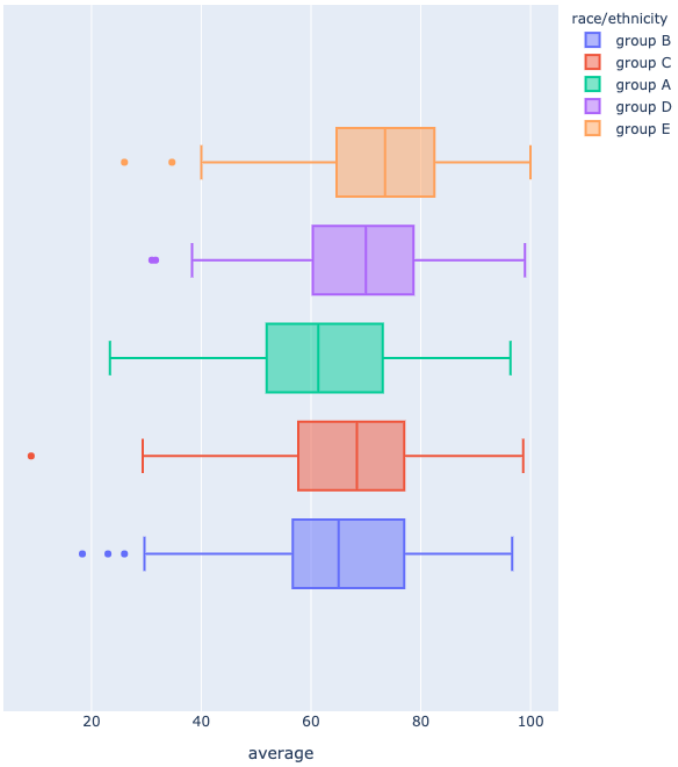


Figure 3: Boxplot comparing the average score between different ethnicities

The above boxplot shows the different medians and quartiles for the different ethnicities present in the dataset, there is a severe overlap and most of the distributions are similar except for group “A” which is substantially lower in terms of minimum, Q1, Q3 and median. This does not provide much predictive power since Group A only represents 8.9% of the overall population (see feature exploration).

Next, we will explore gender in a boxplot below

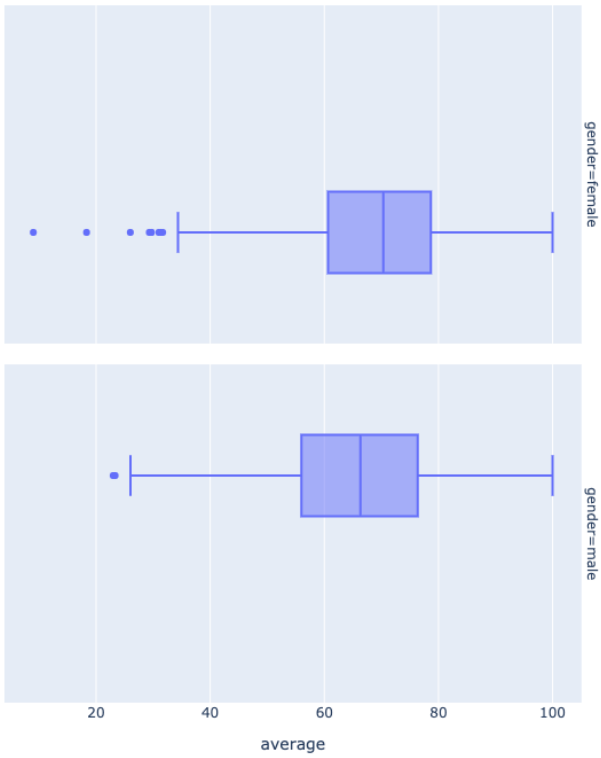


Figure 4: Boxplot showing the difference between male and female

There is not too much difference between the two however the female distribution has a slightly higher Q1, Q3 and median and also has a tighter distribution than the male distribution which is more spread out. The low difference between the two will not lead to high predictive power for this feature.

Test preparation should have a large effect on student performance and is shown in a boxplot figure below.

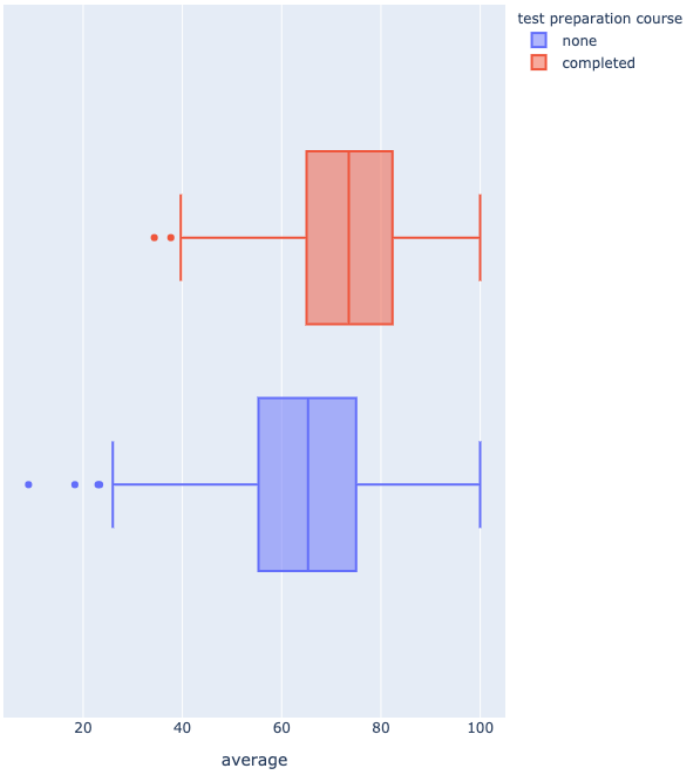


Figure 5: Boxplot comparing the average score of learners who completed test preparation and those who did none

As expected, test preparation has a clear positive effect on learners average score between the tests and will be a good indicator as to whether a student is performing above or below average.

The level of parental education is considered next in a boxplot below

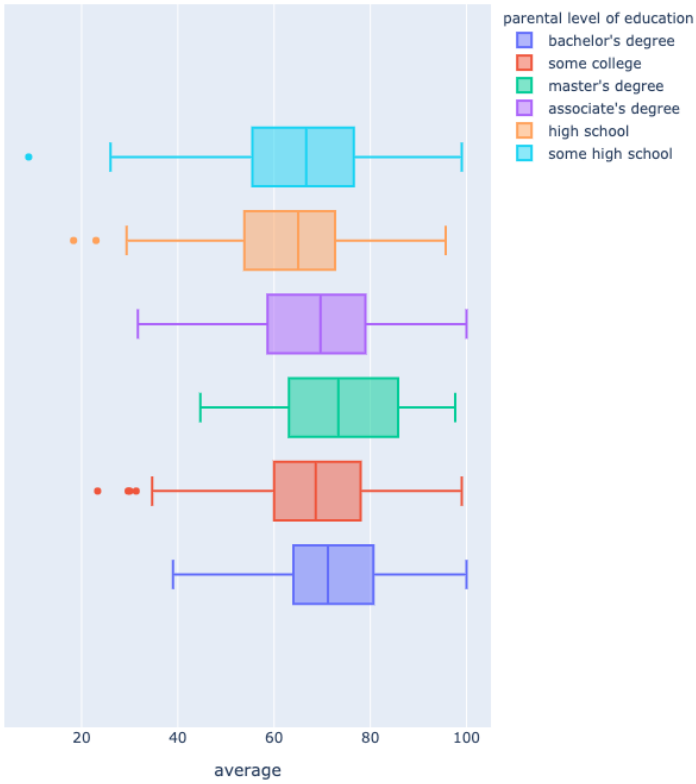


Figure 6: Boxplot showing the distributions for different levels of parental education with regards to the average score

“Bachelors” (dark blue) and “masters” (green) are a cut above the rest and “associates degree”, “some college”, “high school”, “some high-school” are rather difficult to distinguish from each other. It is rather difficult to tell the difference between the six but perhaps a comparison between masters and bachelors combined versus the rest would present a more defined difference.

Trying to combine two of the features into one comparison (below) shows that while the lunch category provides the most separation between itself and parental education level, the education level provides the extra edge to provide extra definition between the lower scoring and higher scoring students. This is promising for later in the modelling stage where all categorical features are used to try and predict students in need of academic assistance.

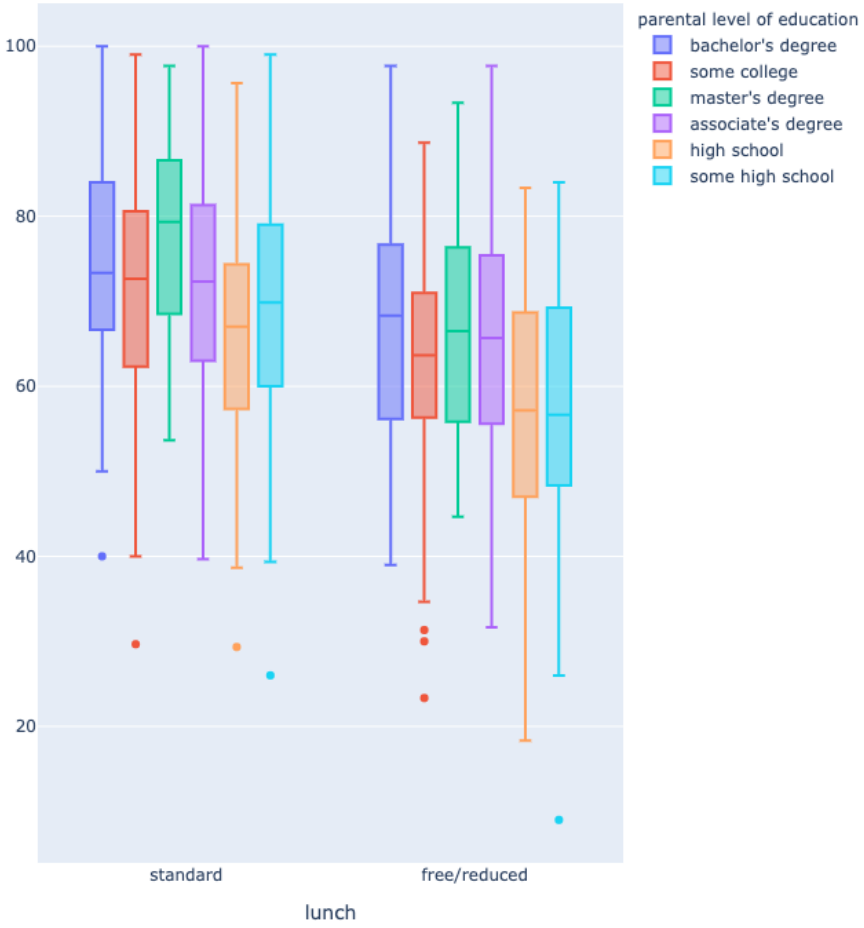


Figure 7: boxplot comparing the distributions between the two lunch categories and the parental level of education

Modelling

The following is a description of the modelling process, for the full procedure see the attached notebook.

After the dummy coding process mentioned in data preparation, three models are chosen. These models are linear regression, Gaussian Naïve Bayes and K Nearest Neighbors. These models were chosen for the following reasons:

Linear regression because the classification task is not too complicated, just requiring it to choose between two possible categories.
Gaussian Naïve Bayes due to its flexibility with classification because the input features are only categorical.
K Nearest Neighbors due to the low number of classes required to predict should make KNN efficient and high performance.

Because the test set is 20% of the total dataset there are a total of 200 entries in the test set, our two prediction categories essentially split the distribution in half, and we can expect around 100 positive and negative cases to be present in our two categories. Those categories being: Below average (0) and Above average (1). The models are asked to predict the full test set against which the performance metrics are calculated for each model and each model also predicts the 10 random samples from the test set whose. The result is then compared to the real answer in the output below the notebook cell.

Evaluation

The three models are evaluated using three metrics, accuracy, precision, and recall.
Accuracy represents the ratio of correct predictions out of all observations.
Precision represents how correct the predictions made truly are, this is the true positive rate.
The recall represents the ratio as to how many members of a class were correctly identified.

The three metrics for the three models are tabulated below:

	Accuracy	Precision	Recall
Naïve Bayes	66.50%	67.04%	66.50%
Linear regression	64.00%	64.02%	64.00%
KNN	59.50%	59.50%	59.50%

The highest score for each metric has been highlighted, Naïve Bayes scores the highest across the board and is the best performing algorithm for this dataset and problem. Linear regression comes in a close second and maybe with a larger dataset or a slightly tweaked problem statement might become the better model.

The confusion matrices are labelled below:

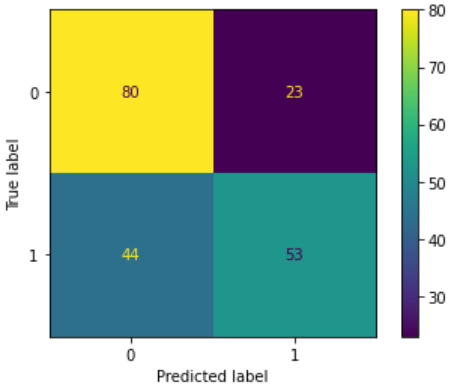


Figure 8: Naive Bayes confusion matrix

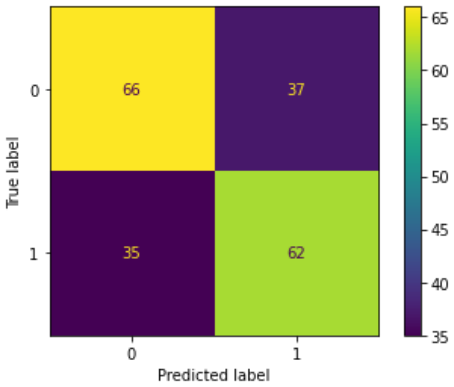


Figure 9: Linear regression confusion matrix

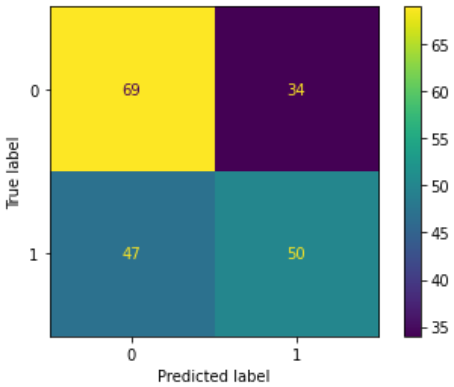


Figure 10: KNN confusion matrix

Judging from the confusion matrices, Naïve Bayes excels at correctly identifying poorly performing students with high precision in the 0^{th} class, however, it also misses classifies almost half of above-average students as below average which is poor performance.

Linear regression is not biased to either of the classes as Naïve Bayes was and has very similar accuracy in both classes with a correct prediction to false prediction ratio of around 2:1. It performs therefore remarkably better in identifying students who are above average and can be seen as the most balanced algorithm.

KNN performs similarly well to Linear regression in identifying below-average students, though not as well as Naïve Bayes. It suffers from the same poor performance in identifying above-average students as Naïve Bayes and is not as good as Linear regression, it, unfortunately, holds the worst of both models of Naïve Bayes and linear regression and is not a recommended solution.

Overall, the performance across the board is not very accurate or precise, this is down to a dataset with not a lot of entries (only 1000), only categorical input features and not a large separation between individual student performance to begin with. The model of choice would not be the highest-scoring (Naïve Bayes) but instead the close second (Linear regression) due to its far more balanced classification of the two classes with almost no penalty in performance versus naïve Bayes.
Should the school or education department decide that rather than looking for students performing below average that they are instead looking to identify students at risk of failing the models may perform better. This is because the separation line between the two classes

we are trying to predict is also the densest point of the distribution, at the median. This means that any slight mistake on the part of the algorithm can easily tip its result over to the wrong class. Should the boundary not coincide with the densest part of the distribution there is still some slight room for error and can increase the performance of the model(s).

PostBlockAssignment3

April 29, 2021

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import sklearn
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
import sklearn.metrics
from sklearn.linear_model import LinearRegression,LogisticRegression
from sklearn.naive_bayes import GaussianNB
from matplotlib import pyplot
```

[2]:
```
def without_hue(plot, feature):
    total = len(feature)
    for p in plot.patches:
        percentage = '{:.1f}%'.format(100 * p.get_height()/total)
        x = p.get_x() + p.get_width() / 2 - 0.05
        y = p.get_y() + p.get_height()
        ax.annotate(percentage, (x, y), size = 12)
    plt.show()
```

[3]:
```
student = pd.read_csv('/Users/jeandre/DataScience874/PostBlockAssignment3/
↪StudentsPerformance.csv')
```

[4]:
```
student.head(5)
```

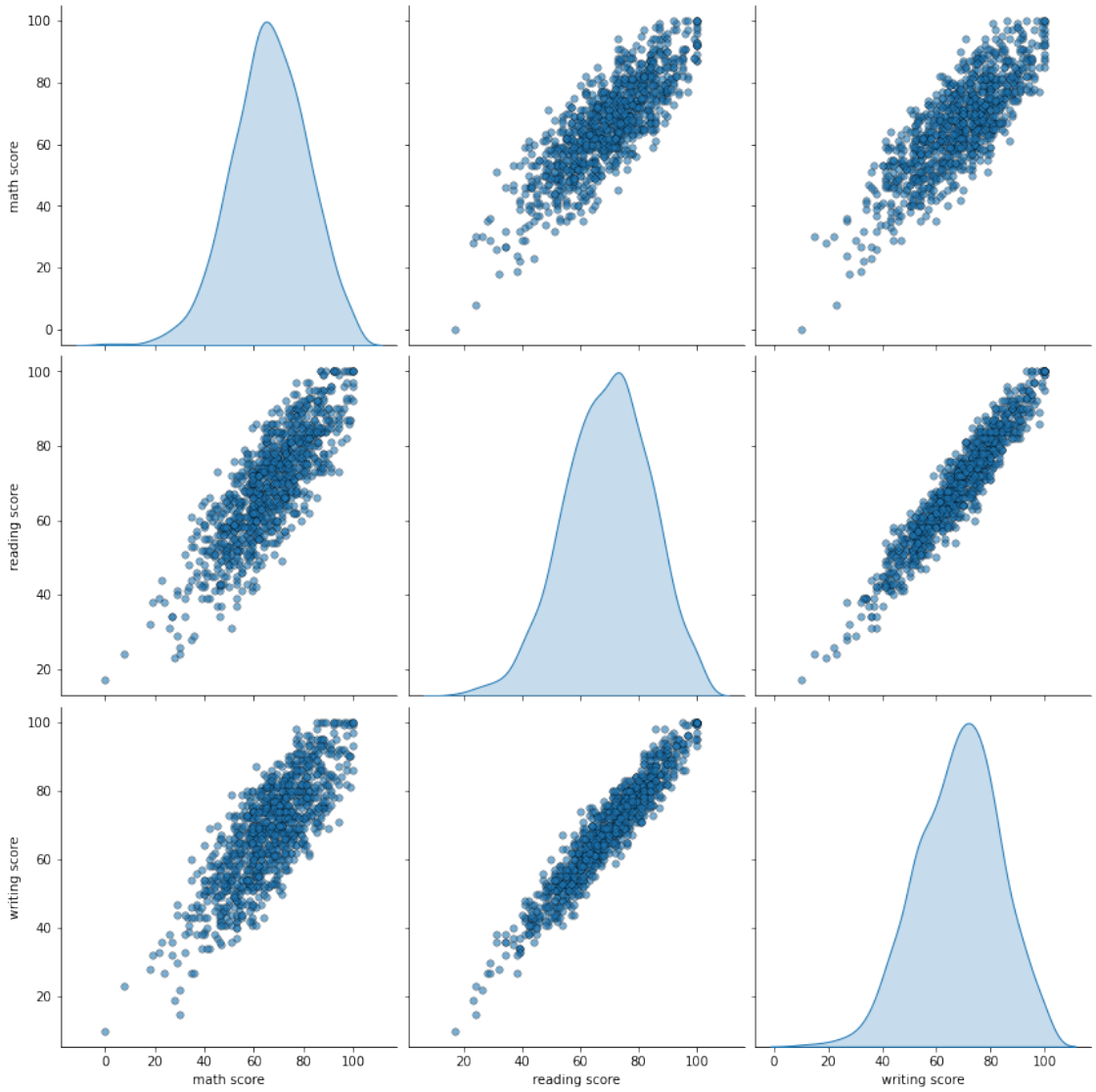
[4]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none			

```
4                    none            76              78              75
```

[5]:
```
sns.pairplot(student, diag_kind = 'kde',
             plot_kws = {'alpha': 0.6, 's': 30, 'edgecolor': 'k'},height = 4)
```

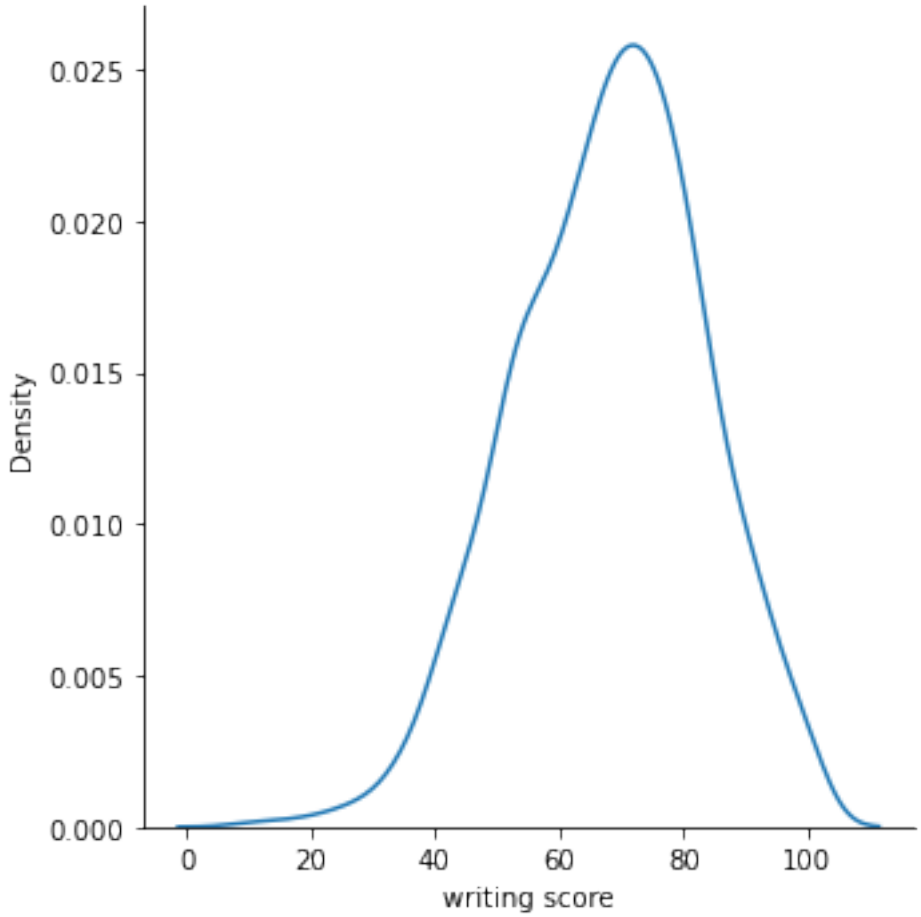
[5]: <seaborn.axisgrid.PairGrid at 0x7faaec68bf70>



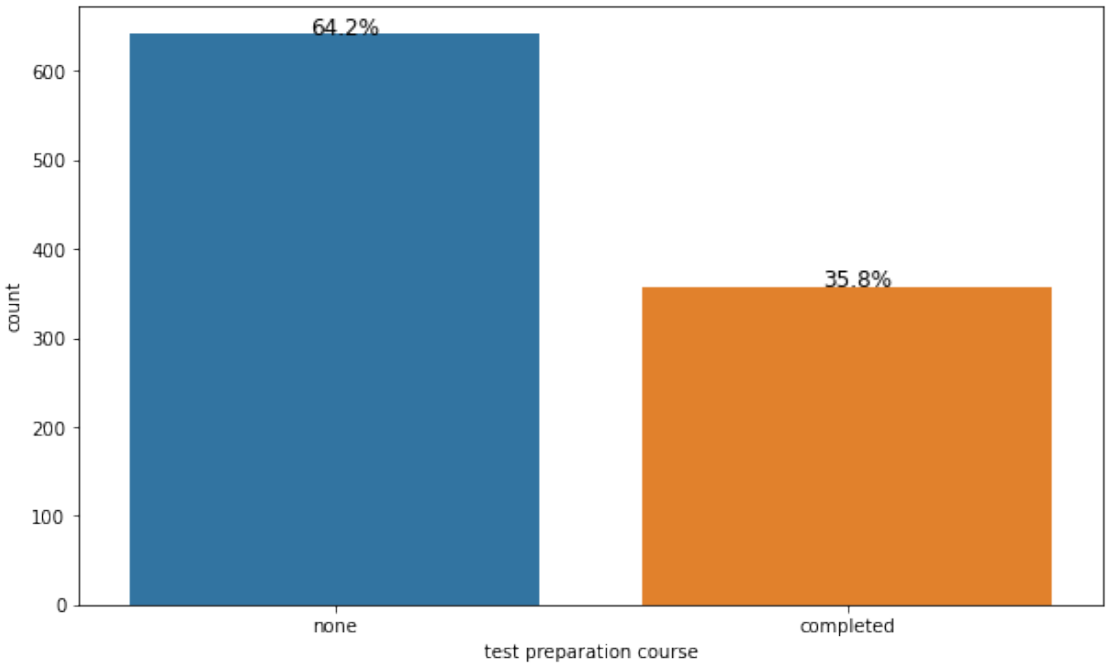
Very strong correlation between the three scores

[6]:
```
sns.displot(data = student, x = "writing score", kind = "kde" )
```

[6]: <seaborn.axisgrid.FacetGrid at 0x7faae886a250>



[7]:
```
dims = (10,6)
fig, ax = pyplot.subplots(figsize=dims)
sns.countplot(data = student, x = "test preparation course" )
without_hue(ax,student["test preparation course"])
```



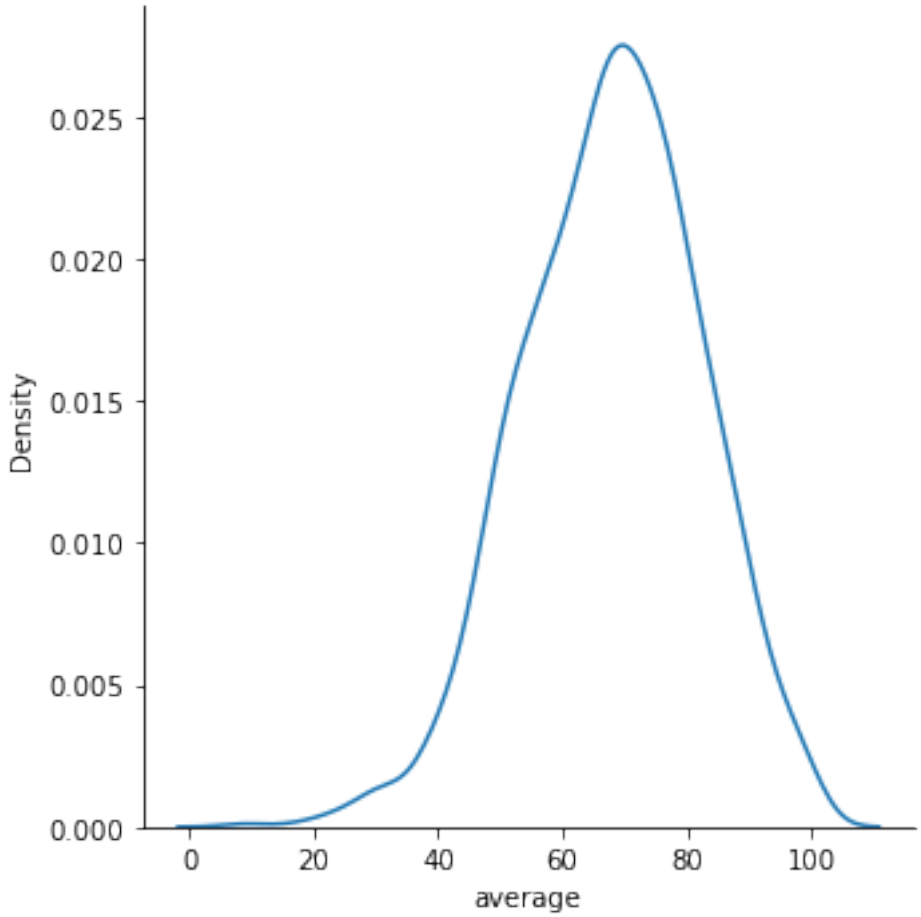
[8]:
```
student['average'] = student.mean(numeric_only=True, axis=1)
student.head(5)
```

[8]:

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	average
0	female	group B	bachelor's degree	standard	none	72	72	74	72.666667
1	female	group C	some college	standard	completed	69	90	88	82.333333
2	female	group B	master's degree	standard	none	90	95	93	92.666667
3	male	group A	associate's degree	free/reduced	none	47	57	44	49.333333
4	male	group C	some college	standard	none	76	78	75	76.333333

[9]:
```
sns.displot(data = student, x = "average", kind = "kde" ) # distribution of␣
↪"average feature"
```

[9]:
```
<seaborn.axisgrid.FacetGrid at 0x7faaeca67b20>
```



[10]:
```
dummy_gender = pd.get_dummies(student['gender'],drop_first=True)
dummy_race = pd.get_dummies(student['race/ethnicity'],drop_first=True)
dummy_education = pd.get_dummies(student['parental level of␣
↪education'],drop_first=True)
dummy_lunch = pd.get_dummies(student['lunch'],drop_first=True)
dummy_test = pd.get_dummies(student['test preparation course'],drop_first=True)
```

0.1 Making dummy coding for the categorical input features

[11]:
```
X = dummy_gender.merge(dummy_race,left_index=True, right_index=True)
X = X.merge(dummy_education,left_index=True, right_index=True)
X = X.merge(dummy_lunch,left_index=True, right_index=True)
X = X.merge(dummy_test,left_index=True, right_index=True)
X.head(2)
```

[11]:
```
   male  group B  group C  group D  group E  bachelor's degree  high school  \
0     0        1        0        0        0                  1            0
1     0        0        1        0        0                  0            0

   master's degree  some college  some high school  standard  none
0                0             0                 0         1     1
1                0             1                 0         1     0
```

binning the avergae score above and below the value of 68. ### 68 is the median and mean of the avergae score.

[12]:
```
Y = pd.cut(student['average'],bins=[0,68,100],labels=[0,1])
Y.head(4)
```

[12]:
```
0    1
1    1
2    1
3    0
Name: average, dtype: category
Categories (2, int64): [0 < 1]
```

[13]:
```
labels = [0,1] #making labels for confusion matrix plotting
```

[14]:
```
X_train, X_test, Y_train, Y_test = sklearn.model_selection.
↪train_test_split(X,Y, test_size = 0.2, random_state = 0)
```

Adding the 10 randomly selected test samples to a prediction set

[15]:
```
prediction_set = []
answer_set = []
for i in range(10):
    x = np.random.randint(200)
    print("entry added = ",x)
    prediction_set.append(np.asarray(X_test.iloc[[x]]))
    answer_set.append(np.asarray(Y_test.iloc[[x]]))
```

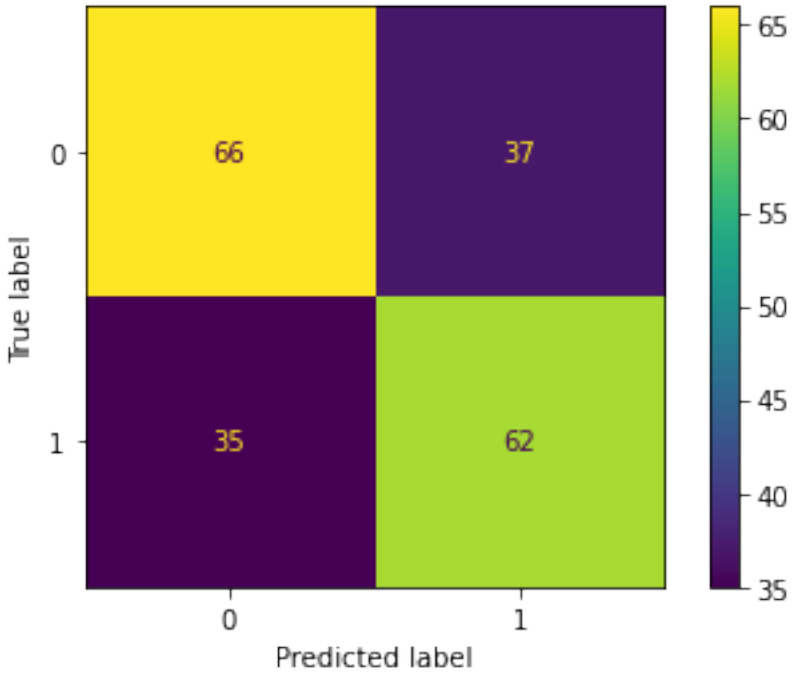
```
entry added =  139
entry added =  100
entry added =  47
entry added =  136
entry added =  134
entry added =  147
entry added =  127
entry added =  188
entry added =  92
entry added =  96
```

0.2 Linear Regression

[16]:
```
results = []
linreg = LogisticRegression().fit(X_train, Y_train)
R_pred = linreg.predict(X_test)
print("Accuracy score =",sklearn.metrics.accuracy_score(Y_test,R_pred))
print("Precision score =",sklearn.metrics.
 ↪precision_score(Y_test,R_pred,average='weighted'))
print("Recall = ",sklearn.metrics.
 ↪recall_score(Y_test,R_pred,average='weighted'))
print(" \n10 Predicted values: ")
for i in range(len(prediction_set)):
    print(answer_set[i][0],"<-answer : predicted->",linreg.
 ↪predict(prediction_set[i]))
sklearn.metrics.
 ↪plot_confusion_matrix(linreg,X_test,Y_test,display_labels=labels)
```

```
Accuracy score = 0.64
Precision score = 0.6402720272027203
Recall =  0.64

10 Predicted values:
1 <-answer : predicted-> [1]
0 <-answer : predicted-> [1]
0 <-answer : predicted-> [0]
0 <-answer : predicted-> [1]
1 <-answer : predicted-> [1]
0 <-answer : predicted-> [0]
1 <-answer : predicted-> [1]
1 <-answer : predicted-> [0]
0 <-answer : predicted-> [0]
0 <-answer : predicted-> [0]
```

[16]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7faaed0b4d00>
```



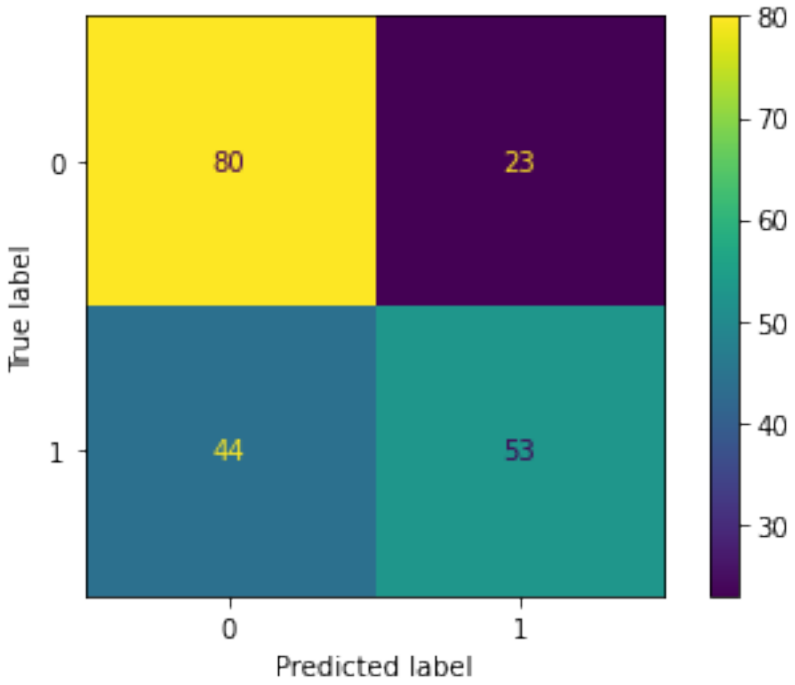
0.3 Gaussian Naive Bayes

[17]:
```
results = []
nb = GaussianNB()
nb.fit(X_train,Y_train)
N_pred = nb.predict(X_test)
print("Accuracy score =",sklearn.metrics.accuracy_score(Y_test,N_pred))
print("Precision score =",sklearn.metrics.
 ↪precision_score(Y_test,N_pred,average='weighted'))
print("Recall = ",sklearn.metrics.
 ↪recall_score(Y_test,N_pred,average='weighted'))

print(" \n10 Predicted values: ")
for i in range(len(prediction_set)):
    print(answer_set[i][0],"<-answer : predicted->",nb.
 ↪predict(prediction_set[i]))

sklearn.metrics.plot_confusion_matrix(nb,X_test,Y_test,display_labels=labels)
```

```
Accuracy score = 0.665
Precision score = 0.6704817487266552
Recall =  0.665
```

```
10 Predicted values:
1 <-answer : predicted-> [0]
0 <-answer : predicted-> [0]
0 <-answer : predicted-> [0]
0 <-answer : predicted-> [1]
1 <-answer : predicted-> [1]
0 <-answer : predicted-> [0]
1 <-answer : predicted-> [1]
1 <-answer : predicted-> [1]
0 <-answer : predicted-> [0]
0 <-answer : predicted-> [0]
```

[17]: `<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7faaed0b4280>`



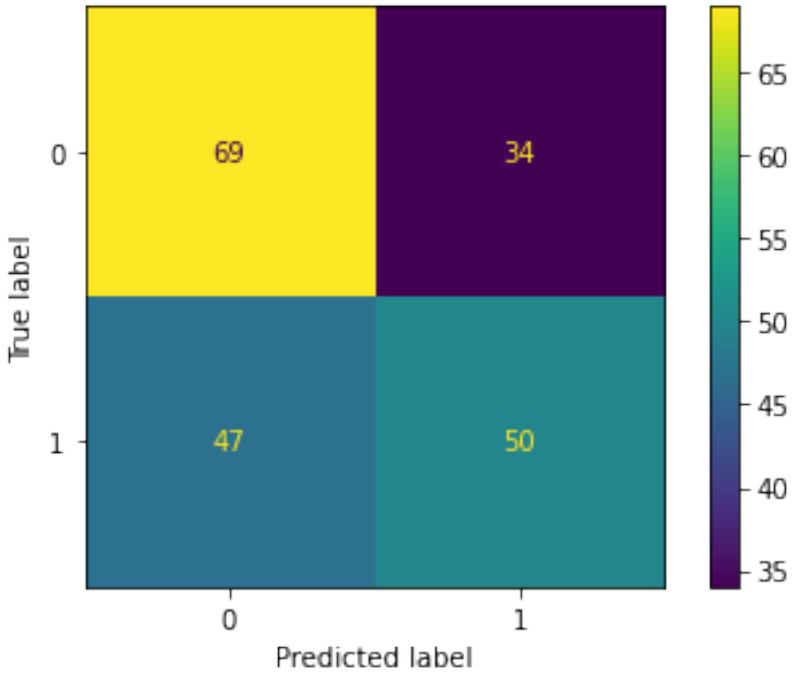
0.4 KNN

[18]:
```
results = []
k = 8
neigh = KNeighborsClassifier(n_neighbors=k)
neigh.fit(X_train,Y_train)
K_pred = neigh.predict(X_test)
print("Accuracy score =",sklearn.metrics.accuracy_score(Y_test,K_pred))
```

```
print("Precision score =",sklearn.metrics.
 ↪precision_score(Y_test,K_pred,average='weighted'))
print("Recall = ",sklearn.metrics.
 ↪recall_score(Y_test,K_pred,average='weighted'))

print(" \n10 Predicted values: ")
for i in range(len(prediction_set)):
    print(answer_set[i][0],"<-answer : predicted->",neigh.
 ↪predict(prediction_set[i]))

sklearn.metrics.plot_confusion_matrix(neigh,X_test,Y_test,display_labels=labels)
```

```
Accuracy score = 0.595
Precision score = 0.595026683087028
Recall =  0.595

10 Predicted values:
1 <-answer : predicted-> [0]
0 <-answer : predicted-> [0]
0 <-answer : predicted-> [0]
0 <-answer : predicted-> [1]
1 <-answer : predicted-> [1]
0 <-answer : predicted-> [0]
1 <-answer : predicted-> [1]
1 <-answer : predicted-> [0]
0 <-answer : predicted-> [0]
0 <-answer : predicted-> [0]
```

[18]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7faaec9bcfa0>


69
34
47
50
0
1
True label
0
1
Predicted label
65
60
55
50
45
40
35

[ ]: