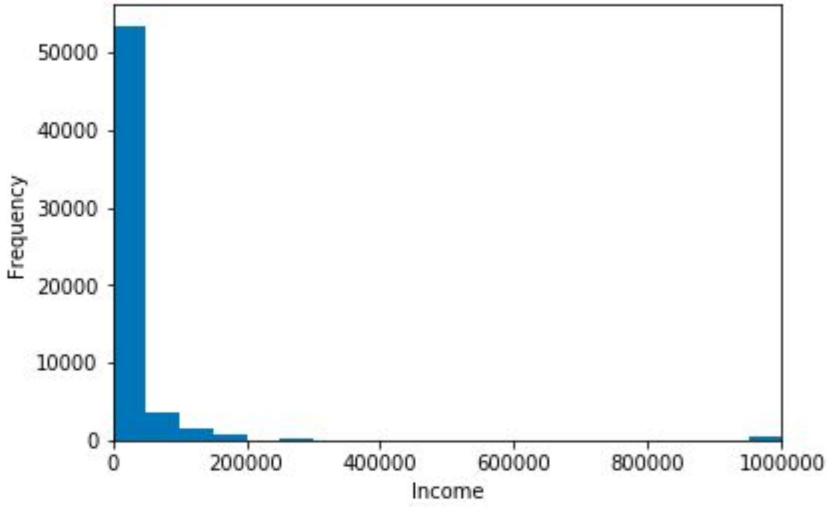
Capstone project: Machine learning fundamentals intensive

Ewout Meijer

Exploration of the dataset - Income Distribution

```
plt.hist(df.income, bins=20)
plt.xlabel("Income")
plt.ylabel("Frequency")
plt.xlim(0, 1000000)
plt.show()
```



Exploration of the dataset - Status (preparation)

In [11]:
```
print(df.status.value_counts())
```

```
single            55697
seeing someone     2064
available          1865
married             310
unknown              10
Name: status, dtype: int64
```

In [12]:
```
total = 55697 + 2064 + 1865 + 310 + 10
```

In [13]:
```
print(total)
```

```
59946
```

In [15]:
```
single_percentage = 55697 / total
seeing_someone_percentage = 2064 / total
available_percentage = 1865 / total
married_percentage = 310 / total
unknown_percentage = 10 / total
```

In [16]:
```
print(single_percentage)
```

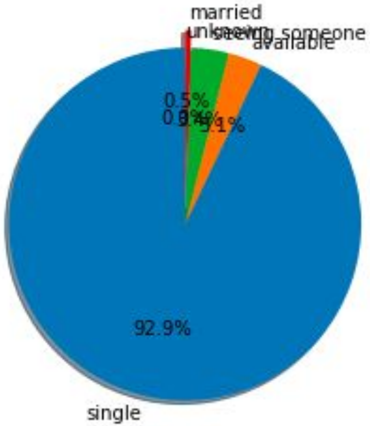
```
0.9291195409201615
```

Exploration of the dataset - Status

```
labels = 'single','available', 'seeing someone', 'married', 'unknown'
sizes = [single_percentage, available_percentage, seeing_someone_percentage, married_percentage, unknown_percentage]


fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')

plt.show()
```



Question statement

Question: Can one’s education and one’s strength of religious believe predict one’s income level?

Reason: I want to see if the one’s moral code (coming from religious believes), and education level have an impact on how well you do financially.

Explanation of new data columns - Education

Column: Education

How I did it: I manually ranked all options and assigned values to them. Then I mapped the values to the dataset.

```
education_mapping = {"dropped out of space camp": 0, "working on space camp": 1, "space camp": 1, "graduated from space
                     "dropped out of high school": 0, "working on high school": 3, "high school": 3, "graduated from hi
                    "dropped out of two-year college": 4, "working on two-year college": 5, "two-year college": 5, "gra
                    "dropped out of college/university": 6, "working on college/university": 7, "college/university": 7
                    "dropped out of masters program": 8, "working on masters program": 9, "masters program": 9, "gradua
"dropped out of law school": 8, "working on law school": 9, "law school": 9, "graduated from law school": 10,
"dropped out of med school": 8, "working on med school": 9, "med school": 9, "graduated from med school": 10,
"dropped out of ph.d program": 10, "working on ph.d program": 11, "ph.d program": 11, "ph.d program": 12}
```

```
all_data["education_code"] = all_data.education.map(education_mapping)
```

Explanation of two new data columns - Religion

Column: Religion

How I did it: I manually ranked all options and assigned values to them. Then I mapped the values to the dataset. For most options in religion this is similar. For agnostics I ranked it form 0.2 - 1. For atheism I ranked it from 0.4 - 0.

```
religion_mapping = {"catholicism and very serious about it": 5, "catholicism and somewhat serious about it":4, "catholic
                    "christianity and very serious about it": 5, "christianity and somewhat serious about it":4, "chris
                    "atheism and very serious about it": 0, "atheism and somewhat serious about it":0.1, "atheism": 0.2
                    "agnosticism and somewhat serious about it":0.8, "agnosticism": 0.6, "agnosticism but not too serio
                    "other and very serious about it": 5, "other and somewhat serious about it":4, "other": 3, "other b
                    "judaism and very serious about it": 5, "judaism and somewhat serious about it":4, "judaism": 3, "j
                    "buddhism and very serious about it": 5, "buddhism and somewhat serious about it":4, "buddhism": 3,
                    "hinduism and very serious about it": 5, "hinduism and somewhat serious about it":4, "hinduism": 3,
                    "islam and very serious about it": 5, "islam and somewhat serious about it":4, "islam": 3, "islam b
```

```
df["religion_code"] = df.religion.map(religion_mapping)
```

Additional features

When running the models it became clear that the predictions improved when more features were included, so I also chose to include age, and whether someone smokes, drinks or does drugs.

Instead of showing the results for each approach with the two dataset, I will only include the calculations where the extended set of features was used.

Drinks:

```
drink_mapping = {"not at all": 0, "rarely": 1, "socially": 2, "often": 3, "very often": 4, "desperately": 5}
```

```
df["drinks_code"] = df.drinks.map(drink_mapping)
```

Drugs:

```
drugs_mapping = {"never": 0, "sometimes": 1, "ofter": 2}
```

```
df["drugs_code"] = df.drugs.map(drugs_mapping)
```

Smokes:

```
smokes_mapping = {"no": 0, "sometimes": 1, "when drinking": 2, "trying to quit": 3, "yes": 4,}
```

```
df["smokes_code"] = df.smokes.map(smokes_mapping)
```

Data preparation:

Not all respondents had filled in their income, so those rows of data had to be removed. The data also needed to be normalized.

Removing income not fileld in:

```
df = df[(df[['income']] != -1).all(axis=1)]
```

Min-Max scaling:

```
feature_data = df[['smokes_code', 'drinks_code', 'drugs_code', 'religion_code', 'education_code']]
```

```
df = df.dropna()
```

```
x = feature_data.values
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)


feature_data = pd.DataFrame(x_scaled, columns=feature_data.columns)
```

Creating the dataset

```
x2 = df[['religion_code','education_code', 'smokes_code', 'drugs_code', 'drinks_code', 'age']]
y2 = df[['income']]
```

```
x2_train, x2_test, y2_train, y2_test = train_test_split(x2, y2, train_size = 0.8, test_size = 0.2, random_state=6)
```

Model 1: K-Nearest Neighbors Classifier

For the classifier I chose to use the K-Nearest Neighbors classifier. The Support Vector Machine did not fit the problem, as it is not a binary problem, and as far as I understand the decision boundary of a SVM is always for binary problems.

Creating the model

```
classifier = KNeighborsClassifier(n_neighbors = 13)
```

Training the model with dataset 1

```
classifier.fit(x2_train, y2_train)
```

Model 1: K-Nearest Neighbors Classifier - timing

Measuring timing with dataset 2

```
total = 0

for i in range(100):

  start = timeit.default_timer()
  accuracies = []
  for k in range(1, 101):
    classifier = KNeighborsClassifier(n_neighbors = k)
    classifier.fit(x2_train, y2_train)
    accuracies_classifier.append(classifier.score(x2_test, y2_test))
  stop = timeit.default_timer()
  total += stop-start

average = total/100

print("Average Runtime: ", end='')
print(average)
```

```
Average Runtime: 0.545990979115013
```

Model 1: K-Nearest Neighbors Classifier - Accuracy

Measuring accuracy with dataset 2:

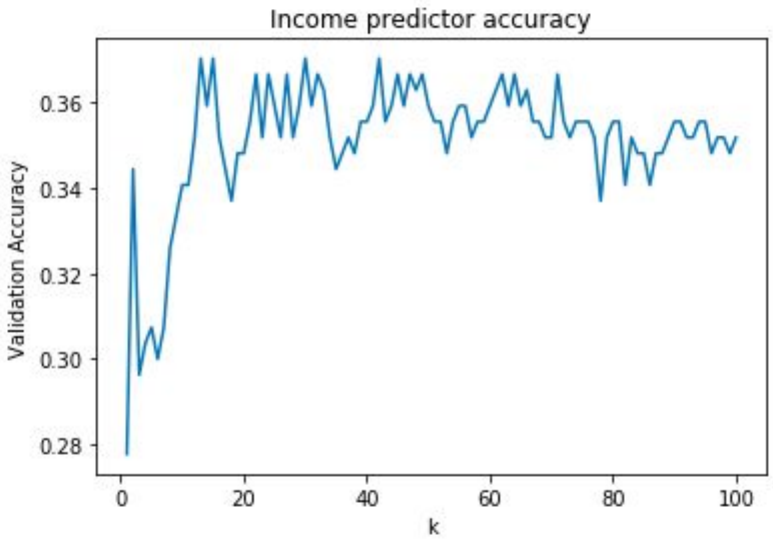
```
accuracies_classifier = []
for k in range(1, 101):
  classifier = KNeighborsClassifier(n_neighbors = k)
  classifier.fit(x2_train, y2_train)
  accuracies_classifier.append(classifier.score(x2_test, y2_test))

k_list = range(1, 101)
print(accuracies_classifier)
plt.plot(k_list, accuracies_classifier)
plt.xlabel("k")
plt.ylabel("Validation Accuracy")
plt.title("Income predictor accuracy")
plt.show
```

```
[0.2777777777777778, 0.3444444444444444, 0.2962962962962963, 0.3037037037037037, 0.3074074074074074, 0.3, 0.30740740
74074074, 0.32592592592592595, 0.3333333333333333, 0.34074074074074073, 0.34074074074074073, 0.35185185185185186, 0.3
7037037037037035, 0.3592592592592593, 0.37037037037037035, 0.35185185185185186, 0.3444444444444444, 0.33703703703703
7, 0.34814814814814815, 0.34814814814814815, 0.35555555555555557, 0.36666666666666664, 0.35185185185185186, 0.3666666
666666664, 0.3592592592592593, 0.35185185185185186, 0.36666666666666664, 0.35185185185185186, 0.3592592592592593, 0.
37037037037037035, 0.3592592592592593, 0.36666666666666664, 0.362962962962963, 0.35185185185185186, 0.34444444444444
44, 0.34814814814814815, 0.35185185185185186, 0.34814814814814815, 0.35555555555555557, 0.35555555555555557, 0.359259
2592592593, 0.37037037037037035, 0.35555555555555557, 0.3592592592592593, 0.36666666666666664, 0.3592592592592593, 0.
36666666666666664, 0.362962962962963, 0.36666666666666664, 0.3592592592592593, 0.35555555555555557, 0.35555555555555
57, 0.34814814814814815, 0.35555555555555557, 0.3592592592592593, 0.3592592592592593, 0.35185185185185186, 0.35555555
555555557, 0.35555555555555557, 0.3592592592592593, 0.362962962962963, 0.36666666666666664, 0.3592592592592593, 0.366
66666666666664, 0.3592592592592593, 0.362962962962963, 0.35555555555555557, 0.35555555555555557, 0.35185185185185186,
0.35185185185185186, 0.36666666666666664, 0.35555555555555557, 0.35185185185185186, 0.35555555555555557, 0.3555555555
5555557, 0.35555555555555557, 0.35185185185185186, 0.337037037037037, 0.35185185185185186, 0.35555555555555557, 0.355
55555555555557, 0.34074074074074073, 0.35185185185185186, 0.34814814814814815, 0.34814814814814815, 0.340740740740740
73, 0.34814814814814815, 0.34814814814814815, 0.35185185185185186, 0.35555555555555557, 0.35555555555555557, 0.351851
85185185186, 0.35185185185185186, 0.35555555555555557, 0.35555555555555557, 0.34814814814814815, 0.35185185185185186,
0.35185185185185186, 0.34814814814814815, 0.35185185185185186]
```

Model 1: K-Nearest Neighbors Classifier - Accuracy

<function matplotlib.pyplot.show(*args, **kw)>



Model 1: KNN Classifier - Precision & Recall

Measuring precision & recall for KNN Classifier:

```
y_pred = classifier.predict(x2_test)
print(confusion_matrix(y2_test, y_pred))
print(classification_report(y2_test, y_pred))
```

```
[[78  2  0  0  0  0  1  7  0  0  0  0]
 [17  0  0  0  0  0  0  6  0  0  0  0]
 [15  1  0  1  0  0  0 11  0  0  0  0]
 [13  0  0  0  0  0  0 14  0  0  0  0]
 [10  1  0  0  0  0  0  5  0  0  0  0]
 [ 5  0  0  0  0  0  0 12  0  0  0  0]
 [10  1  0  0  0  0  0  8  0  0  0  0]
 [ 4  1  1  1  1  0  1 17  0  0  0  0]
 [ 2  1  0  0  0  0  0  5  0  0  0  0]
 [ 1  0  0  0  1  0  0  3  0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0]
 [ 7  1  0  0  0  0  0  4  0  0  0  0]]
             precision    recall  f1-score   support

      20000       0.48      0.89      0.62        88
      30000       0.00      0.00      0.00        23
      40000       0.00      0.00      0.00        28
      50000       0.00      0.00      0.00        27
      60000       0.00      0.00      0.00        16
      70000       0.00      0.00      0.00        17
      80000       0.00      0.00      0.00        19
     100000       0.18      0.65      0.29        26
     150000       0.00      0.00      0.00         8
     250000       0.00      0.00      0.00         5
     500000       0.00      0.00      0.00         1
    1000000       0.00      0.00      0.00        12

avg / total       0.17      0.35      0.23       270
```

Model 2: Multi Linear Regression (MLR)

For the first regression technique I chose to use the Multi Linear Regression.

Training the model with dataset 2

```
model.fit(x2_train, y2_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Measuring accuracy with dataset 2:

```
print(model.score(x2_train, y2_train))
print(model.score(x2_test, y2_test))
```

```
0.004025283800698776
0.006872555120848612
```

Model 2: Multi Linear Regression - timing

Measuring timing with dataset 2

```
total = 0

for i in range(100):

  start = timeit.default_timer()
  y_predict = model.predict(x2_test)
  stop = timeit.default_timer()
  total += stop-start

average = total/100

print("Average Runtime: ", end='')
print(average)
```

```
Average Runtime: 0.00015675136935897172
```

Model 2: Multi Linear Regression - Score

Measuring accuracy with dataset 2:

```
print(model.score(x2_train, y2_train))
print(model.score(x2_test, y2_test))
```

```
0.004025283800698776
0.006872555120848612
```

Model 2: MLR - Precision & Recall

Measuring these metrics is not possible for a linear regression model.

Model 3: K-Nearest Neighbors Regressor

For the second regression technique I chose to use the K-Nearest Neighbors regressor.

```
model2 = KNeighborsRegressor(n_neighbors = k)
model2.fit(x2_train, y2_train)
```

Model 3: K-Nearest Neighbors Regressor - timing

Measuring timing with dataset 2

```
total = 0

for i in range(100):

  start = timeit.default_timer()
  accuracies = []
  for k in range(1, 101):
    classifier = KNeighborsClassifier(n_neighbors = k)
    classifier.fit(x2_train, y2_train)
    accuracies_classifier.append(classifier.score(x2_test, y2_test))
  stop = timeit.default_timer()
  total += stop-start

average = total/100

print("Average Runtime: ", end='')
print(average)
```

```
Average Runtime: 0.545990979115013
```

Model 3: K-NN Regressor - Accuracy, recall & precision

These metrics are only for classifiers and can't be reported for this model.

The score for the model is:

```
print(model2.score(x2_test, y2_test))
```

```
-0.0023569494878559194
```

Comparison of regression models

The Multi Linear regression model had a better score than the K-Nearest Neighbors Regressor. It was also simpler to implement and a lot faster. The K-Nearest Neighbors Classifier proved to be the most accurate, and although it was a lot slower, the predictive capabilities were a lot higher which makes it acceptable.

Because in this case the income was actually categorical data it becomes more difficult to use regression, and it makes sense that the classifier model would outperform the regression models.

Conclusion

Can one’s education and one’s strength of religious believe predict one’s income level?

No, using just these two features is not a good predictor of one’s income. Additional features were added (smoking, drinking, drug use and age), and although they improved the accuracy of the models, the models were still not accurate enough to use for predictions.

Next steps could be to create more numerical data to create additional features.

What other data you would like to have in order to better answer your question(s)?

I would like to have the actual jobs, rather than the sectors. Both a CEO and a doorman can work in the same sector, so the current division isn’t useful. Also the actual salaries rather than the current buckets would be helpful. Although it is unlikely that people are completely honest about their salaries when it might impact their chances of finding a match.