codecademy

Date-A-Scientist Project

Machine Learning Capstone Project

Kaustubh Joshi

April 2020

Table of Contents

Exploring the data

We have a rich dataset consisting of 30 columns and a total of 59946 entries.

Data set currently has three types of data

1. Numerical : Age, Height, Income,
2. Categorical : All the other coolumns
3. Text : essay0-essay9

Some of the categorical data such as that available for drinks, drugs, education, smokes (see sample screenshots below) can be converted to ordinal data to be able to use it for analysis.

```
socially         41780
rarely            5957
often             5164
not at all        3267
very often         471
desperately        322
Name: drinks, dtype: int64
```

```
never          37724
sometimes       7732
often            410
Name: drugs, dtype: int64
```

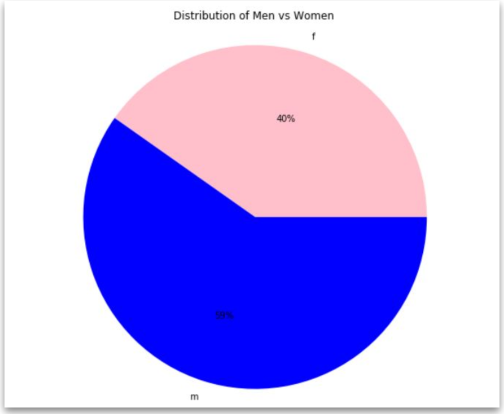
```
RangeIndex: 59946 entries, 0 to 59945
Data columns (total 31 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   age          59946 non-null  int64
 1   body_type    54650 non-null  object
 2   diet         35551 non-null  object
 3   drinks       56961 non-null  object
 4   drugs        45866 non-null  object
 5   education    53318 non-null  object
 6   essay0       54458 non-null  object
 7   essay1       52374 non-null  object
 8   essay2       50308 non-null  object
 9   essay3       48470 non-null  object
 10  essay4       49409 non-null  object
 11  essay5       49096 non-null  object
 12  essay6       46175 non-null  object
 13  essay7       47495 non-null  object
 14  essay8       40721 non-null  object
 15  essay9       47343 non-null  object
 16  ethnicity    54266 non-null  object
 17  height       59943 non-null  float64
 18  income       59946 non-null  int64
 19  job          51748 non-null  object
 20  last_online  59946 non-null  object
 21  location     59946 non-null  object
 22  offspring    24385 non-null  object
 23  orientation  59946 non-null  object
 24  pets         40025 non-null  object
 25  religion     39720 non-null  object
 26  sex          59946 non-null  object
 27  sign         48890 non-null  object
 28  smokes       54434 non-null  object
 29  speaks       59896 non-null  object
 30  status       59946 non-null  object
dtypes: float64(1), int64(2), object(28)
```

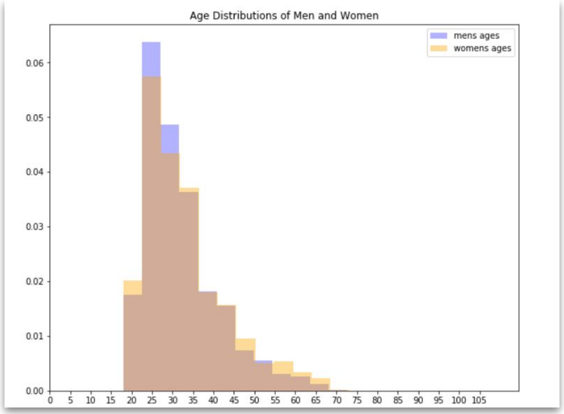
Exploring the data

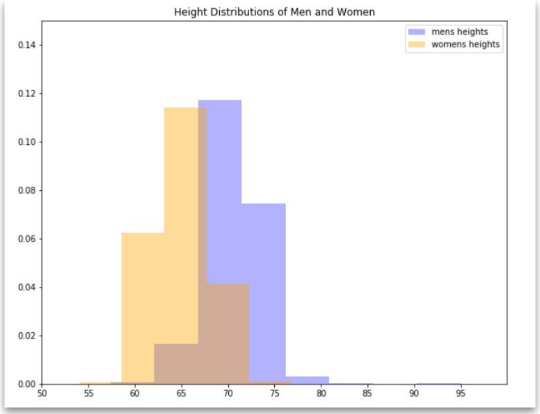
The data set contains a larger proportion of men as compared to Women.

Age distribution matches closely across men as well as Women.

Height distribution for men and women is quite different shown by the histogram



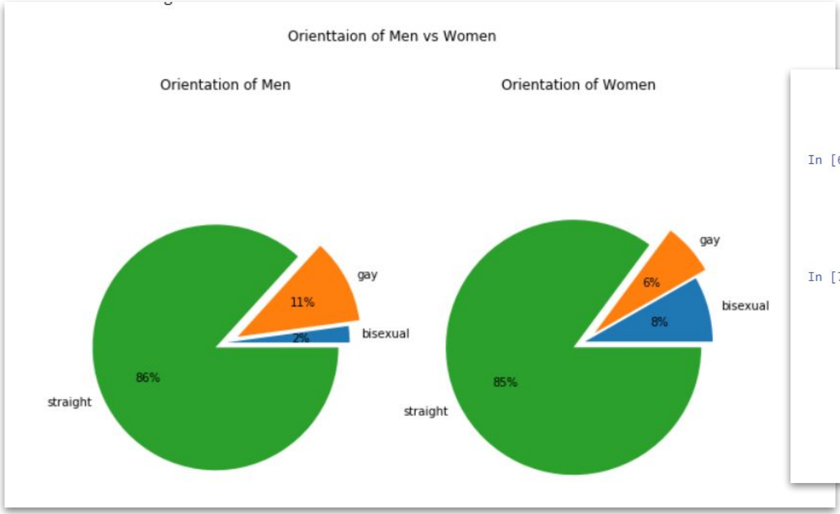




Men vs Women - Sexual preferences differ?

Interestingly, the sexual orientations of men and Women llook quite different.

On running a Chi-Square test, we can see statistical significance for those differences pertining to proportion of bisexual and gay orientation




```
orientation sex  bisexual   gay  straight
0             f      1996  1588     20533
1             m       771  3985     31073
```

```
In [6]: orientation_extract = orientation_pivot[['bisexual', 'gay', 'straight']]
        print(orientation_extract)
```

```
orientation  bisexual   gay  straight
0                1996  1588     20533
1                 771  3985     31073
```

```
In [7]: contingency = orientation_extract.values.tolist()
        print(contingency)
        print(type(contingency))

        from scipy.stats import chi2_contingency

        chi, p, dof, exp = chi2_contingency(contingency)
        print(p)
```

```
[[1996, 1588, 20533], [771, 3985, 31073]]
<class 'list'>
0.0
```

P Value shows that the differences in the sexual orientations are statistically significant

Augmenting data (1)

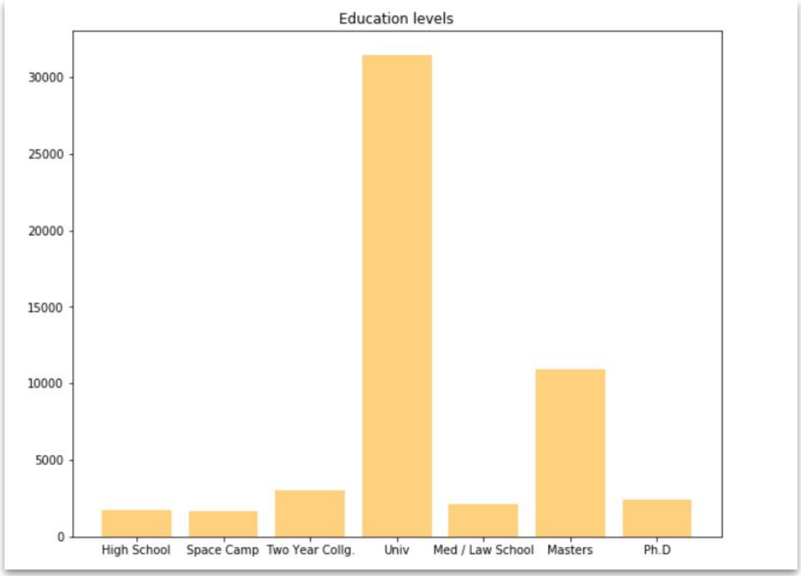
The code besides creates the following additional colums for analysis. These columns aim at converting the categorical data to numerical values

- profiles['sex_int'] (Converting M / F to 1 and)
- profiles['orientation_int']
- profiles['drinks_ordinal']
- profiles['drugs_ordinal']
- profiles['smokes_ordinal']
- Profiles['religion_clean'] (extracting just the religion from the data)
- profiles['religion_int']
- profiles['signs_clean'] (extracting just the sun sign from the signs data)
- profiles['languages_count'] (How many languages does a user know?)

```
profiles['sex_int'] = profiles['sex'].map({'f' : 0, 'm' : 1})

profiles['orientation_int'] = profiles['orientation'].map({'straight' : 0, 'bisexual' : 1, 'gay' : 2})

profiles['drinks_ordinal'] = profiles['drinks'].map({'not at all' : 0, 'rarely' : 1, 'socially' : 2, 'often' : 3, 'very often' : 4, 'desperately' : 5})

profiles['drugs_ordinal'] = profiles['drugs'].map({'never' : 0, 'sometimes' : 1, 'often' : 2})

profiles['smokes_ordinal'] = profiles['smokes'].map({'no' : 0, 'when drinking' : 1, 'sometimes' : 2, 'trying to quit' : 3, 'yes' : 4})

religion_temp = profiles['religion'].str.split(' ')
profiles['religion_clean'] = religion_temp.str[0]

profiles['religion_int'] = profiles['religion_clean'].map({'agnosticism' : 0, 'other' : 1, 'atheism' : 2, 'christianity' : 3, 'catholicism' : 4, 'judaism' : 5, 'buddhism' : 6, 'hinduism' : 7, 'islam' : 8})

sign_temp = profiles['sign'].str.split(' ')
profiles['signs_clean'] = sign_temp.str[0]
profiles['languages_count'] = (profiles['speaks'].str.count(',')) + 1
```

Augmenting data (2)

The code besides helps to convert the education data into a numerical with a new column **profiles['education_clean']** (with PHD being level 7 and HIgh School being Level 1)




```
def findedu(x) :
    phd = 'ph.d' in x
    masters = 'masters' in x
    law_school = 'law school' in x
    med_school = 'med school' in x
    university = 'university' in x
    two_year = 'two-year' in x
    space_camp = 'space camp' in x
    high_school = 'high school' in x

    if phd == True :
        return 7
    elif masters == True :
        return 6
    elif law_school == True :
        return 5
    elif med_school == True :
        return 5
    elif university == True :
        return 4
    elif two_year == True :
        return 3
    elif space_camp == True :
        return 2
    elif high_school == True :
        return 1

profiles['education_clean'] =
profiles['education'].apply(lambda x : findedu(str(x)))
```

Augmenting data (3)

The code besides helps to convert the religion data into a Yes / No column **profiles[‘religious_seriousness’]**

With this column being used as a label, we will Try to predict if we can determine if a person takes religion seriously based on their drinking, smoking, drugs, age and other factors.

```
def how_serious(x) :
    laughing = 'laughing' in x
    not_too_serious = 'not too' in x
    somewhat_serious = 'somewhat' in x
    very_serious = 'very serious' in x

    if laughing == True :
        return 'No'
    elif not_too_serious == True :
        return 'No'
    elif somewhat_serious == True :
        return 'Yes'
    elif very_serious == True :
        return 'Yes'
    elif x == 'nan' :
        return float('nan')
    else :
        return 'No'

profiles['religious_seriousness'] =
profiles['religion'].apply(lambda x : how_serious(str(x)))
```

Classification | Question 1

Can we predict if someone takes religious faith seriously, based on Gender, Orientation, Education, drinking, drugs and smoking habits?

Since the required question has Categorical data, we will avoid the use of KNearestNeighbors, LogisticRegression which depend upon the distances between the points.

Hence we have chosen RandomForest as our method.

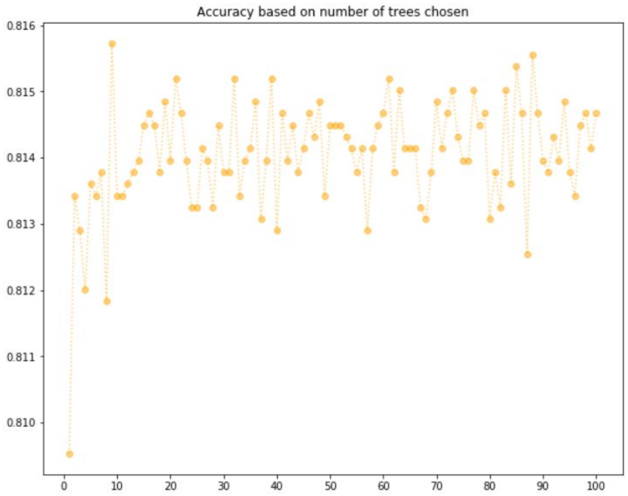
The code besides was used to create the dataframe, create the training and test sets and then implement the RandomForestClassifier.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, recall_score,
precision_score, f1_score

rf_df = profiles.dropna(subset = ['sex_int', 'orientation_int',
'education_clean', 'drinks_ordinal', 'drugs_ordinal', 'smokes_ordinal',
'religion_clean']).reset_index()
#print(rf_df.info())

train_data, test_data, train_labels, test_labels =
train_test_split(rf_df[['sex_int', 'orientation_int', 'drinks_ordinal',
'drugs_ordinal', 'religion_int']], rf_df['religious_seriousness'],
train_size = 0.8, test_size = 0.2, random_state = 1)

scores = []
trees = list(range(1,101))

for i in range (1,101) :
    classifier = RandomForestClassifier(n_estimators = i)
    classifier.fit(train_data, train_labels)
    predictions = classifier.predict(test_data)
    score = classifier.score(test_data, test_labels)
    scores.append(score)
```

Classification | Question 1

Can we predict if someone takes religious faith seriously, based on Gender, Orientation, Education, drinking, drugs and smoking habits?

After looping through different values of n_estimators, we find the number highest **accuracy to be higher than 81.5%**
However the model is **not a reliable model with low recall score (3%) and precision score (38%)**




```
#Score improvements based on number of random trees chosen

plt.close('all')
plt.figure(figsize = (10,8))
ax5 = plt.subplot(1,1,1)
plt.plot(trees, scores, alpha = 0.5, color = 'orange', linestyle = ':',
marker = 'o')
plt.title('Accuracy based on number of trees chosen')
ax5.set_xticks(range(0,101,10))
plt.show()

print(recall_score(test_labels, predictions, pos_label = 'Yes'))
print(precision_score(test_labels, predictions, pos_label = "Yes"))
```

Classification | Question 2

Can we predict sex based on content in essays?

Since it is expected that men and women would have rather different likes, dislikes and ways of writing, we want to see if we can predict sex based on the content in essays.

This model results in an **accuracy of 73%** and an overall **f1_score of 70%**

The model is **moderately successful** in predicting sex based on the text in essays.

```
nb_df =
profiles[["essay0","essay1","essay2","essay3","essay4","essay5","essay6",
"essay7","essay8","essay9",'sex']]
nb_df = nb_df.replace(np.nan, '', regex = True)
nb_df['essays_all'] =
nb_df[["essay0","essay1","essay2","essay3","essay4","essay5","essay6","es
say7","essay8","essay9"]].apply(lambda row : ' '.join(row), axis = 1)

train_data, test_data, train_labels, test_labels =
train_test_split(nb_df['essays_all'], nb_df['sex'], train_size = 0.8,
test_size = 0.2, random_state = 1)

counter = CountVectorizer()
counter.fit(train_data)
train_counts = counter.transform(train_data)
test_counts = counter.transform(test_data)

classifier = MultinomialNB()
classifier.fit(train_counts, train_labels)
predictions = classifier.predict(test_counts)
score_nb = classifier.score(test_counts, test_labels)

print(score_nb)
```

Regression | Question 1

Can we predict income based on length of essay and avg word length

STEP 1 : Preparing the DataFrame

The code besides was used to prepare the dataframe, and create additional columns for length of essays, word count and average word length

```
ml_df =
profiles[["essay0","essay1","essay2","essay3","essay4","essay5","essay6",
"essay7","essay8","essay9",'income', 'age']]

#Filling in empty essays with ''
ml_df.fillna(value = {'essay0' : '', 'essay1' : '','essay2' : '','essay3'
: '','essay4' : '','essay5' : '','essay6' : '','essay7' : '','essay8' :
'','essay9' : ''}, inplace = True)

#Joining all essays together and then finding length, word count and avg
word length
ml_df['essays_all'] =
ml_df[["essay0","essay1","essay2","essay3","essay4","essay5","essay6","es
say7","essay8","essay9"]].apply(lambda row : ' '.join(row), axis = 1)
ml_df['essays_len'] = ml_df['essays_all'].apply(lambda x : len(x))
ml_df['word_count'] = ml_df['essays_all'].str.count(' ') + 1
ml_df['avg_word_len'] = ml_df['essays_len'] / ml_df['word_count']
ml_df['freq_i_me'] = ml_df['essays_all'].str.count('\s[iI]\s|\s[mM]e\s')

print(ml_df.info())
print(ml_df['essays_all'].head())
print(ml_df['freq_i_me'].head())
```

Regression | Question 1

Can we predict income based on length of essay and avg word length

STEP 2 : Train Test Split, Normalisation

The code besides was used to create the data frame with columns and rows relevant for our prediction model

1. SInce a lot of rows contained income = ‘-1’, they will be counterproductive to our analysis.
   Therefore we reduce our data set to extract such rows.\
2. Then we split our resultant dataset into training and test datasets
3. Then we normalise our dataset using Standard Scalar from sklearn.preprocessing

```
#Prepare the DF
multi_df = ml_df[['essays_len', 'avg_word_len','income']]
multi_df = multi_df[multi_df['income'] > 10].reset_index(drop=True)
print(multi_df.info())

#Prepare Train and Test Data
x_train, x_test, y_train, y_test =
train_test_split(multi_df[['essays_len', 'avg_word_len']],
multi_df['income'], train_size = 0.8, test_size = 0.2, random_state = 1)

#Normalising the data

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
x_scaled_train = scaler.fit_transform(x_train)
x_scaled_test = scaler.transform(x_test)

print(x_scaled_train)
print(x_scaled_test)
```

Regression | Question 1

Can we predict income based on length of essay and avg word length

STEP 3A : Creating the Regression Model (Linear Regression)

The code besides was used to create the object, fit the data and then analyse scores using MultipleLinearRegression

We see that the **scores are extremely low, and the model cannot be used for prediction.**

```
from sklearn.linear_model import LinearRegression

my_model = LinearRegression()
my_model.fit(x_scaled_train, y_train)
guesses = my_model.predict(x_scaled_test)
print(my_model.score(x_scaled_train, y_train))
print(my_model.score(x_scaled_test, y_test))

#Scores are extremely low and hence length of essay and wordcount cannot
be used to predict income.

0.003907412945515332
0.0013597515420864514
```

Regression | Question 1

Can we predict income based on length of essay and avg word length

STEP 3B : Creating the Regression Model (KNearestNeighbors)

The code besides was used to create the object, fit the data and then analyse scores using KNearestNeighbors

We see that again, **scores are extremely low, and the model cannot be used for prediction.**

```
#Trying the scores using K Nearest neighbors.

from sklearn.neighbors import KNeighborsRegressor

my_model = KNeighborsRegressor(n_neighbors = 5, weights = 'uniform')
my_model.fit(x_scaled_train, y_train)
print(my_model.score(x_scaled_test, y_test))

-0.17951139282140738
```

Regression | Question 2

Can we predict age based on frequency of I, Me in the essays?

The code besides was used to

1. Cout the number of ‘I’ and ‘Me’ in the essays
2. Create the Traing & Test datasets
3. Normalise the data
4. Create the model and analyse scores.

We see that again, **scores are extremely low, and the model cannot be used for prediction.**

```
#Adding a column to count Frequency of I & me
ml_df['freq_i_me'] = ml_df['essays_all'].str.count('\s[iI]\s|\s[mM]e\s')

#Train & Test Data
x = np.array(ml_df['freq_i_me']).reshape(-1,1)
print(x)

x_train, x_test, y_train, y_test = train_test_split(x, ml_df['age'],
train_size = 0.8, test_size = 0.2, random_state = 1)

#Normalising the data

scaler = StandardScaler()
x_scaled_train = scaler.fit_transform(x_train)
x_scaled_test = scaler.transform(x_test)


my_model = LinearRegression()
my_model.fit(x_scaled_train, y_train)
print(my_model.score(x_scaled_test, y_test))

#Scores are extremely low and linear regression cannot be applied to
predict age based on the frequency of 'I' or me in essays
0.0036377087137188235
```

Conclusion

Through this project we can conclude the following :

1. We have statistical significance to prove that sexual preferences of women are much different as compared to those of men, with a much higher bisexual orientation in women
2. We were able to use Random forests to predict with a moderately high accuracy if a person takes religion seriously based on their orientation and habits. The recall score and precision score on this were however low, which makes the model unusable
3. We were able to use Naive Bayes Model to predict sex based on content in essays with a 73% accuracy and a similar f1_score, indicating it to be moderately usable
4. We were unable to use essays data to predict income, or age based on content in essays despite using multiple regression models.