Classification of MSDS Students' Relationship Status

Prince Joseph Erneszer Javier
MSc Data Science
Asian Institute of Management

Executive Summary

An online survey was conducted on Asian Institute of Management's MSc Data Science students. 42 students were asked a total of 46 questions, one of which is the student's relationship status broken down into: not in a relationship, in a relationship, or in a complicated relationship.

The problem being tackled here is whether a machine learning model will be able to predict with good accuracy a student's relationship status based on the 45 other questions. Three models were evaluated: k Nearest Neighbors (kNN) Classifier, Logistic Regression, and Linear Support Vector Classification (SVC)

As expected, the single best predictor for whether or not a person is in a relationship, not in a relationship, or in a complicated relationship is `How many times in a month do you go out with your s.o`. The second best predictor is `How many partners have you had?`. It might be possible that those students in a relationship had strong affinity to be in a relationship. The sole student in a complicated relationship have had four partners. The third predictor is `How many movies do you see inside a theater in a year?` (kNN) or `Number of movies you watched from cinema in the last two weeks` (logistic regression and Linear SVC). Looking back at the histograms, we see that in general, those not in a relationship watch movies in theatres more frequently than those in relationships. Might it be possible that those not in a relationship are actually still on the dating phase, watching movies with their dates?

The top three features that give the highest accuracy when taken individually are shown in the table below.

kNN Accuracy	Logistic Regression Accuracy	Linear SVC Accuracy
How many times in a month do you go out with your s.o.?	How many times in a month do you go out with your s.o.?	How many times in a month do you go out with your s.o.?
How many partners have you had?	How many partners have you had?	How many partners have you had?
How many movies do you see inside a theater in a year?	Number of movies you watched from cinema in the last two weeks	Number of movies you watched from cinema in the last two weeks

When only `How many times in a month do you go out with your s.o` was used, Logistic Regression and Linear SVC yielded the highest average classification accuracy. The table below summarizes the accuracies and optimal parameters for the three classification models. The minimum required accuracy is 60% (1.25 X PCC).

Model	Model Parameter	Parameter Value	Maximum Ave. Accuracy
kNN	k	3	0.8963
Logistic Regression	C	0.0205	0.9296
Linear SVC	C	0.0068	0.9111

The results show that using any of the three classification models mentioned above can classify the 42 students based on relationship status with around 90% accuracy.

For a more interesting classification, `How many times in a month do you go out with your s.o` and `How many partners have you had?` were excluded in the next modeling. Using trial and error, features were identified for each model that maximized accuracy. Linear SVC gives the highest performance with 94% accuracy.

Model	Model Parameter	Parameter Value	Maximum Ave. Accuracy	No. of Features
kNN	k	1	0.8481	11
Logistic Regression	C	1.632720	0.9000	9
Linear SVC	C	0.5102	0.9407	13

The features common in all models are:

Features
Number of movies you watched from cinema in the last two weeks
How many siblings are you in the family? (not counting half siblings)
What phone brand do you use? Samsung, Apple, Vivo, Huawei, Sony, etc._samsung
What is your favorite number, from 0 to 9?

Aside from frequency of watching movies in cinemas, another interesting feature is the number of siblings in the family. Those in a relationship have on the average more siblings than those not in a relationship. Might it be possible that the number of siblings somehow affects a person's inclination to be in a relationship, e.g. by developing a family-centric attitude? What about the phone brands and even more strangely, favorite number? These are questions that may be tackled by future research.

Take note that these conclusions may not generalize over other groups of people since the datapoints in this analysis were limited to only 42 MSc Data Science students, which may not be representative of other populations.

Data Description

Data were collected from an online survey of 42 MSc Data Science students at the Asian Institute of Management, Makati, Philippines. There were a total of 46 questions. The answer to question 6, which is whether or not the person is in a relationship is what we tried to classify. The survey was conducted online through Google Forms during class.

The questions were:

1. Number of movies you watched from cinema in the last two weeks
2. Number of movies you watched from home in the last two weeks
3. Average Cups of coffee consumed per day
4. Average amount of bills paid for mobile used
5. Average number of posts in social media per week
6. In a relationship (Yes , No, it's complicated)
7. How old are you?
8. Male or female? (pls type M or F)
9. What is your height in cm?
10. What is your weight in kg?
11. What is your waist size in inches?
12. What is your favorite color?
13. What is your favorite number, from 0 to 9?
14. Are you predominantly right handed or left handed? (pls type R or L)
15. What is your favorite car brand? Toyota, Honda, Hyundai, * BMW, Mercedes, etc.
16. How many cars have you owned in your life?
17. What phone network do you use? Globe, Smart, Sun, etc.
18. What phone brand do you use? Samsung, Apple, Vivo, Huawei, Sony, etc.
19. How many years do you use your phone before replacing it?
20. How many cities have you lived in? (Must have lived more than 1 year)
21. How many siblings are you in the family? (not counting half siblings)
22. What is your favorite sport? Basketball, football, volleyball, tennis, etc.
23. What browser do you prefer? Chrome, IE, Firefox, Safari
24. How many functioning shoes do you own currently?
25. What is your blood type? O A B AB
26. How many countries have you visited?
27. How many provinces in the Philippines have you visited?
28. Where is your ideal city to live in?
29. How many times in a month do you go to the mall?
30. What is your favorite mall? SM, Ayala, Robinsons, Rustan's, etc.
31. What is your favorite fastfood place?
32. What is your favorite meat to eat? Chicken, pork, beef, fish, etc.
33. How many times in a month do you eat in Jollibee?
34. How many times in a week do you cook at home?
35. How many times in a month do you go out with your s.o.?
36. How many movies do you see inside a theater in a year?
37. How many years in your life have you been a smoker?
38. How many beers can you drink in one night?
39. How many pizzas can you eat in one sitting?
40. How many times do you buy coffee in a week?
41. How many Facebook friends do you have? (In hundreds, e.g. 1100)
42. How many partners have you had?
43. What kind of songs do you usually listen to? Pop, hip-hop, R&B, alternative, rock, classical, country, jazz, etc.
44. How many songs do you know the lyrics to?
45. What do you think your grade in this class will be?
46. What age do you think you will live to?

Functions and Packages

Below are the functions and packages that were used for exploratory data analysis and predictive modeling. They were placed into this section first as a single location for all coded functions and second, in order not to crowd the rest of the sections below.

Python Packages

`Numpy` gives a wide array of functionality when dealing with arrays (pun intended). `Matplotlib` allows us to make charts easily. `Pandas` enable us to transform data into tables that are more understandable and transformable.

We also import `Counter` that allows us to quickly count the number of unique values in a sequence of values like a list, array, or dictionary.

Finally, we import four packages from `sklearn`. `Train_test_split` allows us to split samples in a dataset into test set and training set. Training set is what we feed the classification algorithm, while test set is what we use to test the accuracy of the algorithm. The last three packages are `KNeighborsClassifier`, `LogisticRegression`, and `LinearSVC` which are classification algorithms. These three will be discussed in more detail in the succeeding sections.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd

from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
```

Data Exploration Functions

To aid with data exploration, the function `conv_to_numeric` was developed. This function scans through a dataframe and will try to convert the contents to numerical values. If it's unable to convert the contents automatically, it will notify which columns contained non-numeric values and the values that couldn't be converted. This is useful for example if we want to convert "1500" encoded as string or text to the number 1500 or to identify "2000 pesos" as a value that couldn't be converted. We can then manually change it to 2000.

In [2]:
```
# Defining convertion function
def conv_to_numeric(df, inds_0):
    """
    Convert all values in dataframe to numeric if possible, otherwise, skip.
    Print list of values that cannot be converted and must be converted
    manually. Returns new indices of columns that were not completely
    converted.

    Inputs
    ======
    df: DataFrame of values to convert
    inds_0: List of indices of columns in DataFrame to convert
    """

    # Initial pass: convert to float, ignore data if unconvertable
    for i in inds_0:
        df.iloc[:, i] = df.iloc[:, i].astype(float, errors='ignore')

    # Second pass, show the unconvertable data per feature
    inds_1 = []
    for i in inds_0:
        typ = df.iloc[:, i].dtype
        if typ == object:
            inds_1.append(i)

    print("No. of columns that must be changed to numerical:", len(inds_0))
    print("No. of columns automatically changed to numbers, 1st pass:",
          len(inds_0) - len(inds_1))

    print()
    print("Remaining columns with values that must be changed to numbers:")

    if len(inds_1) == 0:
        print("All values numerical already.")

    else:
        index = []
        cols = []
        vals = []
        for i in inds_1:
            v = df.iloc[:, i][np.logical_not(
                (df.iloc[:, i]).str.isnumeric())].values
            index.append(i)
            cols.append(df.columns[i])
            vals.append(v)

        index2 = pd.Series(index)
        cols2 = pd.Series(cols)
        vals2 = pd.Series(vals)

        df2 = pd.concat([index2, cols2, vals2], axis=1)
        df2.columns = ['Index', 'Names', 'Values']
        display(df2)

    return inds_1
```

The function `column_non_num` accepts a dataframe and returns the column indices and names of columns that contain non-numerical values. This is useful when looking for columns that are expected to contain numerical data, but still contains non-numerical data. For example, if a column for weights include "32 kg", this is read by the computer as non-numerical because of " kg". This function will help us more easily generate a list of those columns.

In [3]:
```
def column_non_num(df):
    """
    Accept dataframe and print columns with index of columns containing
    non-numerical data

    Input
    =====
    df: Dataframe with multiple columns

    Returns
    =======
    List of tuples of indices and columns with non-numerical data
    """
    inds = []
    cols = []
    for i in range(len(df.columns)):
        typ = df.iloc[:, i].dtype
        if typ != 'float64' and typ != 'int64':
            inds.append(i)
            cols.append(df.columns[i])
    print("No. of columns containing non-numerical data:", len(inds))

    return list(zip(inds, cols))
```

Classification Models

Below is the function defined for three classification models: k Nearest Neighbors Classifier (kNN), Logistic Regression, and Support Vector Machine (SVM) Classifier. The dataset that will be analyzed is composed of observations or samples with various features and one target variable. The classification models are supervised, which means we know from the start the features and the classifications of the observations we will use to train the models.

The general methodology for "training" the classification algorithms will be as follows: The datapoints/samples/observations in the dataset will be randomized. Then they will be split into two: training set and test set. The training set will be used to train the classification models while the test set, which the models shouldn't have "seen" before will be used to check the models' accuracy. The classification model's objective is to correctly classify a new observation based on its features.

The function below can be used to select one of the three classification models, generate a plot of accuracy vs. model parameters, and output a report showing the maximum average accuracy achieved, the standard deviation of accuracy over multiple iterations, the optimal parameters, and the minimum required accuracy (proportional chance criterion).

In [4]:
```
def ml_class(feature, target, ml_type='knn_class', show_PCC=False,
             param_range=range(1, 30), seed_settings=range(0, 30),
             plot=False, report=True, penalty='l2'):
    """
    Plot accuracy vs parameter for test and training data. Print
    maximum accuracy and corresponding parameter value. Print number of trials.

    Inputs
    ======
    feature: Dataframe of features
    target: Series of target values
    show_PCC: Boolean. will show PCC on plot if True
    param_range: Range of values for parameters
    seed_settings: Range of seed settings to run
    plt: Boolean. Will show plot if True
    report: Boolean. Will show report if True
    penalty: String either l1 for L1 norm or l2 for L2 norm

    Outputs
    =======
    Plot of accuracy vs parameter for test and training data
    Report showing number of maximum accuracy, optimal parameters, PCC, and
        no. of iterations
    """

    train_acc = []
    test_acc = []

    # Initiate counter for number of trials
    iterations = 0

    # create an array of cols: parameters and rows: seeds
    for seed in seed_settings:

        # count one trial
        iterations += 1

        # split data into test and training sets
        X_train, X_test, y_train, y_test = train_test_split(feature,
                                                            target,
                                                            random_state=seed)
        train = []
        test = []
        lasso = []

        # make a list of accuracies for different parameters
        for param in param_range:
            # build the model
            if ml_type == 'knn_class':
                clf = KNeighborsClassifier(n_neighbors=param)

            elif ml_type == 'log_reg':
                clf = LogisticRegression(C=param, penalty=penalty)

            elif ml_type == 'svc':
                clf = LinearSVC(C=param, penalty=penalty)

            clf.fit(X_train, y_train)

            # record training set accuracy
            train.append(clf.score(X_train, y_train))
            # record generalization accuracy
            test.append(clf.score(X_test, y_test))

        # append the list to _acc arrays
        train_acc.append(train)
        test_acc.append(test)

    # compute mean and error across columns
    train_all = np.mean(train_acc, axis=0)
    test_all = np.mean(test_acc, axis=0)

    # compute standard deviation
    var_train = np.var(train_acc, axis=0)
    var_test = np.var(test_acc, axis=0)

    # compute pcc
    state_counts = Counter(target)
    df_state = pd.DataFrame.from_dict(state_counts, orient='index')
    num = (df_state[0] / df_state[0].sum())**2
    pcc = 1.25 * num.sum()

    if plot == True:

        # plot train and errors and standard devs
        plt.plot(param_range, train_all, c='b',
                 label="training set", marker='.')
        plt.fill_between(param_range,
                         train_all + var_train,
                         train_all - var_train,
                         color='b', alpha=0.1)

        # plot test and errors and standard devs
        plt.plot(param_range, test_all, c='r', label="test set", marker='.')
        plt.fill_between(param_range,
                         test_all + var_test,
                         test_all - var_test,
                         color='r', alpha=0.1)

        # plot pcc line
        if show_PCC == True:
```

```
            plt.plot(param_range, [pcc] * len(param_range),
                     c='tab:gray', label="pcc", linestyle='--')

        plt.xlabel('Parameter Value')
        plt.ylabel('Accuracy')
        plt.title(ml_type + ": Accuracy vs Parameter Value")
        plt.legend(loc=0)

        plt.tight_layout()
        plt.show()

    max_inds = np.argsort(test_all)[-1]
    acc_max = test_all[max_inds]
    param_max = list(param_range)[max_inds]

    if report == True:
        print('Report:')
        print('=======')
        print("Max average accuracy: {}".format(
            np.round(acc_max, 4)))
        print("Var of accuracy at optimal parameter: {0:.4f}".format(
            var_test[max_inds]))
        print("Optimal parameter: {0:.4f}".format(param_max))
        print('1.25 x PCC: {0:.4f}'.format(pcc))
        print('Total iterations: {}'.format(iterations))

    # return maximum accuracy and corresponding parameter value
    return np.round(acc_max, 4), param_max
```

Exploratory Data Analysis

Importing Data

The dataset is saved in a comma-separated values (csv) file `ACS-ML Data.csv`. The dataset was loaded in a pandas dataframe, `df`, which organizes the data in table form. Each column in the pandas dataframe corresponds to a feature of an observation or sample. For example, one feature is the average cups of coffee consumed per day. Meanwhile, the rows represent each observation or sample, which in this case, is each student surveyed.

The dataset contains 46 columns and 43 rows or samples. However, take note that there were only 42 student respondents. From this alone, we can infer that the data needs some checking and cleaning before we can run our predictive models. As the adage says, "garbage in, garbage out." A clean data is imperative.

In [5]:
```
df = pd.read_csv('ACS-ML Data.csv')
print("No. of features/columns in the dataset: {}".format(df.shape[1]))
print("No. of samples/rows in the dataset: {}".format(df.shape[0]))
```

```
No. of features/columns in the dataset: 46
No. of samples/rows in the dataset: 43
```

Looking at the first three rows and last three rows, we see that there are entries with `NaN` values

In [6]:
```
cols = [i for i in range(len(df.columns))]
_ = df.copy()
_.columns = cols

print("Showing the first three rows:")
print("==========")
display(_.head(3))

print()
print("Showing the last three rows:")
print("==========")
display(_.tail(3))
```

```
Showing the first three rows:
==========
```

	0	1	2	3	4	5	6	7	8	9	...	36	37	38	39	40	41	42	43	44	45
0	0.0	0.0	0.0	1799	1.0	Yes	30.0	M	168	67	...	0	1	3	0.0	933.0	4.0	EDM	5	5	128.0
1	NaN	NaN	NaN	NaN	NaN	NaN	29.0	F	160	62	...	0	1	3 slices	6.0	700.0	0.0	pop, jazz, rock, classical, R&B	30	4.5	90.0
2	0.0	0.0	0.0	800	0.0	Yes	33.0	M	177	150	...	0	6	4	0.0	800.0	5.0	Rock	10	5	82.0

3 rows × 46 columns

```
Showing the last three rows:
==========
```

	0	1	2	3	4	5	6	7	8	9	...	36	37	38	39	40	41	42	43	44	45
40	0.0	0.0	1.0	1400	1.0	It's complicated	27.0	M	171	67	...	0	5	1	1.0	1627.0	4.0	Country	100	5	100.0
41	0.0	1.0	0.0	3500	4.0	Yes	37.0	M	163	65	...	0	0	0	0.0	400.0	1.0	R&D	3	4.5	90.0
42	1.0	4.0	2.0	900	2.0	Yes	38.0	M	165	68	...	1	4	3	2.0	1150.0	6.0	Pop	5	4.75	70.0

3 rows × 46 columns

Cleaning the Data

All 46 columns contain at least one `NaN` value. The first six columns contain 6 `NaN` values.

In [7]:
```
a = np.sum([df.isnull().any()])
b = list(df.loc[:,df.isnull().any()])

print("No. of columns with null: {}".format(a))
print("==========")

for i in range(len(b)):
    s = np.sum(df.iloc[:,i].isnull())
    print("No. of null values in column {}: {}".format(i, s))
```

```
No. of columns with null: 46
==========
No. of null values in column 0: 6
No. of null values in column 1: 6
No. of null values in column 2: 6
No. of null values in column 3: 6
No. of null values in column 4: 6
No. of null values in column 5: 6
No. of null values in column 6: 1
No. of null values in column 7: 1
No. of null values in column 8: 1
No. of null values in column 9: 1
No. of null values in column 10: 1
No. of null values in column 11: 1
No. of null values in column 12: 1
No. of null values in column 13: 1
No. of null values in column 14: 1
No. of null values in column 15: 1
No. of null values in column 16: 1
No. of null values in column 17: 1
No. of null values in column 18: 1
No. of null values in column 19: 1
No. of null values in column 20: 1
No. of null values in column 21: 1
No. of null values in column 22: 1
No. of null values in column 23: 1
No. of null values in column 24: 1
No. of null values in column 25: 1
No. of null values in column 26: 1
No. of null values in column 27: 1
No. of null values in column 28: 1
No. of null values in column 29: 1
No. of null values in column 30: 1
No. of null values in column 31: 1
No. of null values in column 32: 1
No. of null values in column 33: 1
No. of null values in column 34: 3
No. of null values in column 35: 1
No. of null values in column 36: 1
No. of null values in column 37: 1
No. of null values in column 38: 1
No. of null values in column 39: 1
No. of null values in column 40: 1
No. of null values in column 41: 1
No. of null values in column 42: 1
No. of null values in column 43: 1
No. of null values in column 44: 1
No. of null values in column 45: 1
```

There are 8 rows that have at least one null or `NaN` entry. We drop all samples with `NaN` values as they may skew the prediction. We save the new dataset in `_df`. The new dataset contains 35 samples with 46 features.

In [8]:
```
a = df[df.isnull().any(axis=1)].shape[0]
b = list(df[df.isnull().any(axis=1)].index)

print("No. of rows with null: {}".format(a))
print("Rows with null: {}".format(b))
print("===========")

for i in b:
    s = np.sum(df.iloc[i, :].isnull())
    print("No. of null values in row {}: {}".format(i, s))

_df = df.drop(b, axis='rows')
_df = _df.reset_index(drop=True)


a2 = _df[_df.isnull().any(axis=1)].shape[0]
b2 = list(_df[_df.isnull().any(axis=1)].index)

print("\nAfter dropping rows with NaN:")
print("===========")
print("No. of features/columns in the dataset: {}".format(_df.shape[1]))
print("No. of samples/rows in the dataset: {}".format(_df.shape[0]))
print("No. of rows with null: {}".format(a2))
print("Rows with null: {}".format(b2))

for i in b2:
    s = np.sum(_df.iloc[i, :].isnull())
    print("No. of null values in row {}: {}".format(i, s))
```

```
No. of rows with null: 8
Rows with null: [1, 3, 10, 14, 23, 29, 30, 35]
===========
No. of null values in row 1: 6
No. of null values in row 3: 6
No. of null values in row 10: 6
No. of null values in row 14: 6
No. of null values in row 23: 40
No. of null values in row 29: 1
No. of null values in row 30: 7
No. of null values in row 35: 6

After dropping rows with NaN:
===========
No. of features/columns in the dataset: 46
No. of samples/rows in the dataset: 35
No. of rows with null: 0
Rows with null: []
```

Below are the columns or features that contain non-numerical data. For categorical features like favorite color, the values are expected to be non-numerical. However, some features like height and weight were found to contain non-numerical data.

In [9]:
```
# Do not truncate information in dataframe
pd.set_option('display.max_colwidth', -1)


a = column_non_num(_df)
inds = pd.Series([i for i, j in a], name='Index')
cats = pd.Series([j for i, j in a], name='Feature Name')

_ = pd.concat([inds, cats], axis=1)
display(_)
```

```
No. of columns containing non-numerical data: 30
```

	Index	Feature Name
0	3	Average amount of bills paid for mobile used
1	5	In a relationship (Yes , No, it's complicated)
2	7	Male or female? (pls type M or F)
3	8	What is your height in cm?
4	9	What is your weight in kg?
5	11	What is your favorite color?
6	13	Are you predominantly right handed or left handed? (pls type R or L)
7	14	What is your favorite car brand? Toyota, Honda, Hyundai, BMW, Mercedes, etc.
8	16	What phone network do you use? Globe, Smart, Sun, etc.
9	17	What phone brand do you use? Samsung, Apple, Vivo, Huawei, Sony, etc.
10	18	How many years do you use your phone before replacing it?
11	21	What is your favorite sport? Basketball, football, volleyball, tennis, etc.
12	22	What browser do you prefer? Chrome, IE, Firefox, Safari
13	23	How many functioning shoes do you own currently?
14	24	What is your blood type? O A B AB
15	27	Where is your ideal city to live in?
16	28	How many times in a month do you go to the mall?
17	29	What is your favorite mall? SM, Ayala, Robinsons, Rustan's, etc.
18	30	What is your favorite fastfood place?
19	31	What is your favorite meat to eat? Chicken, pork, beef, fish, etc.
20	32	How many times in a month do you eat in Jollibee?
21	33	How many times in a week do you cook at home?
22	34	How many times in a month do you go out with your s.o.?
23	35	How many movies do you see inside a theater in a year?
24	36	How many years in your life have you been a smoker?
25	37	How many beers can you drink in one night?
26	38	How many pizzas can you eat in one sitting?
27	42	What kind of songs do you usually listen to? Pop, hip-hop, R&B, alternative, rock, classical, country, jazz, etc.
28	43	How many songs do you know the lyrics to?
29	44	What do you think your grade in this class will be?

Numerical data were expected for the following features in the table below. However, upon inspection, it was found that they contain non-numerical data as well. This means that the non-numerical data need to be converted to numerical data.

Index	Feature Name
3	Average amount of bills paid for mobile used
8	What is your height in cm?
9	What is your weight in kg?
18	How many years do you use your phone before replacing it?
23	How many functioning shoes do you own currently?
28	How many times in a month do you go to the mall?
32	How many times in a month do you eat in Jollibee?
33	How many times in a week do you cook at home?
34	How many times in a month do you go out with your s.o.?
35	How many movies do you see inside a theater in a year?
36	How many years in your life have you been a smoker?
37	How many beers can you drink in one night?
38	How many pizzas can you eat in one sitting?
43	How many songs do you know the lyrics to?
44	What do you think your grade in this class will be?

All contents of 3 of the 15 columns or features above were converted automatically to numbers in the first pass of conversion. However, some values in the remaining 12 columns had to be manually converted to numbers.

In [10]:
```
# Index of columns that must be converted to numbers
inds = [3, 8, 9, 18, 23, 28, 32, 33, 34, 35, 36, 37, 38, 43, 44]
inds_1 = conv_to_numeric(_df, inds);
```

```
No. of columns that must be changed to numerical: 15
No. of columns automatically changed to numbers, 1st pass: 3

Remaining columns with values that must be changed to numbers:
```

	Index	Names	Values
0	3	Average amount of bills paid for mobile used	[300.00, 800.00, 1,500 pesos]
1	18	How many years do you use your phone before replacing it?	[3-4, 1.5, 4.5, 1.5]
2	23	How many functioning shoes do you own currently?	[7 pairs]
3	28	How many times in a month do you go to the mall?	[daily, ~20]
4	32	How many times in a month do you eat in Jollibee?	[twice a month]
5	33	How many times in a week do you cook at home?	[daily, twice a month, Once]
6	34	How many times in a month do you go out with your s.o.?	[na]
7	35	How many movies do you see inside a theater in a year?	[<10]
8	36	How many years in your life have you been a smoker?	[na]
9	38	How many pizzas can you eat in one sitting?	[1/3, 0.33]
10	43	How many songs do you know the lyrics to?	[150+, <10, 30+, 50+]
11	44	What do you think your grade in this class will be?	[3.00, 3.5, B+, 4 - 5, 4.25, 5.00, A, 4.0, 4.0, 4.75, 3.25, 4.5, 4..5, 4.5, 4.75]

The values were changed to their numerical values. The following changes were done:

Index	Original Values	New Values
3	'300.00' '800.00' '1,500 pesos'	300.00, 800.00, 1500.00
18	'3-4' '1.5' '4.5' '1.5'	3.5, 1.5, 4.5, 1.5
23	'7 pairs'	7
28	'daily' '~20'	30, 20
32	'twice a month'	2
33	'daily' 'twice a month' 'Once'	7, 0.5, 1
34	'na'	0
35	'<10'	10
36	'na'	0
38	'1/3' '0.33'	0.33, 0.33
43	'150+' '<10' '30+' '50+'	150, 10, 30, 50
44	'3.00' '3.5' 'B+' '4 - 5' '4.25' '5.00' 'A' '4.0' '4.0' '4.75' '3.25' '4.5' '4..5' '4.5' '4.75'	All numbers will be retained, B+ will be 4.5 and A will be 5, '4-5' will be 4.5

The new dataframe was saved in `_df2`.

In [11]:
```
new_val = [[300, 800, 1500], [3.5, 1.5, 4.5, 1.5], [7], [30, 20], [2],
           [7, 0.5, 1], [0], [10], [0], [0.33, 0.33], [150, 10, 30, 50],
           [3.00, 3.5, 4.5, 4.5, 4.25, 5, 5, 4, 4, 4.75, 3.25, 4.5, 4.5, 4.5,
            4.75]]

_df2 = _df.copy()

j = 0
for i in inds_1:
    inds = list(_df2.iloc[:, i][np.logical_not(
        _df2.iloc[:, i].str.isnumeric())].index)
    if inds != []:
        _df2.iloc[inds, i] = new_val[j]
    j += 1
```

In [12]:
```
inds = [3, 8, 9, 18, 23, 28, 32, 33, 34, 35, 36, 37, 38, 43, 44]
conv_to_numeric(_df2, inds);
```

```
No. of columns that must be changed to numerical: 15
No. of columns automatically changed to numbers, 1st pass: 15

Remaining columns with values that must be changed to numbers:
All values numerical already.
```

Out of 46 columns, 15 are categorical or non-numerical. These categories are enumerated below.

In [13]:
```
print("No. of columns in cleaned dataframe: {}\n".format(len(_df2.columns)))
a = column_non_num(_df2)
[print(i, j) for i, j in a];
```

```
No. of columns in cleaned dataframe: 46

No. of columns containing non-numerical data: 15
5 In  a relationship (Yes ,  No,  it’s complicated)
7 Male or female? (pls type M or F)
11 What is your favorite color?
13 Are you predominantly right handed or left handed? (pls type R or L)
14 What is your favorite car brand? Toyota, Honda, Hyundai, BMW, Mercedes, etc.
16 What phone network do you use? Globe, Smart, Sun, etc.
17 What phone brand do you use? Samsung, Apple, Vivo, Huawei, Sony, etc.
21 What is your favorite sport? Basketball, football, volleyball, tennis, etc.
22 What browser do you prefer? Chrome, IE, Firefox, Safari
24 What is your blood type? O A B AB
27 Where is your ideal city to live in?
29 What is your favorite mall? SM, Ayala, Robinsons, Rustan’s, etc.
30 What is your favorite fastfood place?
31 What is your favorite meat to eat? Chicken, pork, beef, fish, etc.
42 What kind of songs do you usually listen to? Pop, hip-hop, R&B, alternative, rock, classical, country, jazz, etc.
```

Columns 5, 13, 17, 21, 27, 30, and 42 contain some values that are similar but were spelled differently. These vaues will need to be cleaned or standardized. Two examples of similar values but spelled differently are "mcdonalds" & "mcdonald's", and "yes" & "yeeeeeessss". The cleaned up dataframe is saved in `_df3`.

In [14]:
```
# lower all categorical values, remove apostrophes, trailing spaces
cat_i = [i for i, j in a]  # indices of categorical variables
for i in cat_i:
    _df2.iloc[:, i] = _df2.iloc[:, i].str.lower()
    _df2.iloc[:, i] = _df2.iloc[:, i].str.strip()
    _df2.iloc[:, i] = _df2.iloc[:, i].str.replace("'", "")

for i in cat_i:
    print("Column Index:",i)
    print("Column Name:",_df2.columns[i])
    vals = _df2.iloc[:, i].value_counts().values
    index = _df2.iloc[:, i].value_counts().index
    b = pd.DataFrame(index, vals).reset_index()
    b.columns = ["Counts", "Answer"]
    display(b)
```

Column Index: 5
Column Name: In a relationship (Yes , No, it’s complicated)

	Counts	Answer
0	18	yes
1	15	no
2	1	yeeeeeessss
3	1	its complicated

Column Index: 7
Column Name: Male or female? (pls type M or F)

	Counts	Answer
0	30	m
1	5	f

Column Index: 11
Column Name: What is your favorite color?

	Counts	Answer
0	17	blue
1	5	black
2	4	green
3	3	red
4	1	pink
5	1	teal
6	1	white
7	1	orange
8	1	gray
9	1	maroon

Column Index: 13
Column Name: Are you predominantly right handed or left handed? (pls type R or L)

	Counts	Answer
0	28	r
1	6	l
2	1	right

Column Index: 14
Column Name: What is your favorite car brand? Toyota, Honda, Hyundai, BMW, Mercedes, etc.

	Counts	Answer
0	10	toyota
1	4	honda
2	3	volvo
3	3	hyundai
4	2	bmw
5	2	lamborghini
6	2	none
7	2	ford
8	1	subaru
9	1	tesla
10	1	mitsubishi
11	1	audi
12	1	jeep
13	1	mazda
14	1	ferarri

Column Index: 16
Column Name: What phone network do you use? Globe, Smart, Sun, etc.

	Counts	Answer
0	29	globe
1	4	smart
2	1	tm
3	1	sun

Column Index: 17
Column Name: What phone brand do you use? Samsung, Apple, Vivo, Huawei, Sony, etc.

	Counts	Answer
0	16	apple
1	9	samsung
2	2	asus
3	2	oppo
4	1	lennovo
5	1	huawei
6	1	google
7	1	lenovo
8	1	oneplus
9	1	nokia

Column Index: 21
Column Name: What is your favorite sport? Basketball, football, volleyball, tennis, etc.

	Counts	Answer
0	8	basketball
1	7	badminton
2	4	table tennis
3	4	volleyball
4	2	crossfit
5	2	boxing
6	1	swimming
7	1	judo
8	1	competitive real time strategy games
9	1	football
10	1	none
11	1	chess
12	1	basketbal
13	1	sleeping

Column Index: 22
Column Name: What browser do you prefer? Chrome, IE, Firefox, Safari

	Counts	Answer
0	31	chrome
1	2	firefox
2	1	safari
3	1	ie

Column Index: 24
Column Name: What is your blood type? O A B AB

	Counts	Answer
0	16	o
1	11	b
2	4	a
3	4	ab

Column Index: 27
Column Name: Where is your ideal city to live in?

	Counts	Answer
0	2	singapore
1	2	paris
2	2	new york
3	1	cavite
4	1	melbourne
5	1	st. gallen
6	1	paranaque
7	1	lipa
8	1	mandaluyong city
9	1	makati
10	1	yakutsk city
11	1	tagbilaran, bohol
12	1	manila
13	1	laguna
14	1	metro manila
15	1	cebu
16	1	taipei
17	1	alabang
18	1	somewhere in europe
19	1	baguio
20	1	polilio
21	1	hometown
22	1	hong kong
23	1	tagaytay
24	1	yokohama
25	1	tagaytay city
26	1	vancouver
27	1	bern
28	1	toronto
29	1	baliuag
30	1	mandaue
31	1	warsaw

Column Index: 29
Column Name: What is your favorite mall? SM, Ayala, Robinsons, Rustan’s, etc.

	Counts	Answer
0	17	ayala
1	10	sm
2	2	greenbelt
3	1	rockwell
4	1	robinsons
5	1	shangri-la
6	1	farmers plaza
7	1	up town center
8	1	mall of asia

Column Index: 30
Column Name: What is your favorite fastfood place?

	Counts	Answer
0	14	jollibee
1	5	mcdonalds
2	4	kfc
3	3	chowking
4	2	tokyo tokyo
5	1	mcdo
6	1	frankies
7	1	andoks
8	1	burger king
9	1	kenny rogers
10	1	none
11	1	wendys

```
Column Index: 31
Column Name: What is your favorite meat to eat? Chicken, pork, beef, fish, etc.
```

	Counts	Answer
0	17	chicken
1	6	beef
2	5	fish
3	5	pork
4	1	spaghetti
5	1	salmon

```
Column Index: 42
Column Name: What kind of songs do you usually listen to? Pop, hip-hop, R&B, alternative, rock, classical, country, jazz, etc.
```

	Counts	Answer
0	7	pop
1	3	country
2	2	indie
3	2	r&b
4	2	rock
5	2	hip-hop
6	2	alternative
7	2	edm
8	1	classical, rock
9	1	acoustic
10	1	r&d
11	1	hip-hop, r&b, alternative, rock, classical, country
12	1	j-rock
13	1	alternative rock
14	1	indie rock
15	1	love
16	1	opm
17	1	alternative, pop, acoustic pop, classical
18	1	death metal
19	1	classical, pop
20	1	jazz, r&b, classical, country, soul

In [15]:
```
pd.options.mode.chained_assignment = None  # Disable false positive warning

# Cleaning

_df3 = _df2.copy()

lst = [5, 13, 17, 21, 27, 30, 42]
old = ["yeeeeeessss", "right", "lennovo", "basketbal", "tagaytay city",
       "mcdo", "r&d"]
new = ["yes", "r", "lenovo", "basketball", "tagaytay", "mcdonalds", "r&b"]

i = 0
for n in lst:
    _df3.iloc[:, n][_df2.iloc[:, n] == old[i]] = new[i]
    i += 1
```

In order to use the classification models, the values in the dataframe have to be numerical. The categorical feature values will be converted to numerical values using `get_dummies` method which converts all unique values into separate "features" or columns which may take only 0 or 1. 0 if the observation/sample/data point does not possess the feature and 1 if it does. This is appropriate because it's not possible to tell which categorical value is better over the other. For example, how can we tell if "blue" is 3 and "green" is 2? On the other hand, the 1s and 0s generated by `get_dummies` merely act like an indicator whether or not the feature applies for a certain data point.

First, we separate the target column (column 5) from the dataframe and save in `y_temp`. Then we split the dataframe of features into numerical feature values and categorical feature values `df_num` and `df_cat` respectively.

We split `songs` from `df_cat` and apply `get_dummies` separately in order to split the song genres in the column using a delimeter of ",". We then apply `get_dummies` on `df_cat` and save to `df_cat2`. Afterwhich, we combine `df_num`, `df_cat2`, and `songs` to a final features dataframe `X`. We don't need to use `get_dummies` for `df_num` because the values contained are already numerical.

In [16]:
```
a = column_non_num(_df3)
y_temp = _df3.iloc[:, 5]
c_inds = [i for i, j in a if i != 5 and i != 42]
n_inds = [i for i in range(len(_df3.columns))
          if i not in c_inds and i != 5 and i != 42]
print("No. of columns prior to dummification:", len(n_inds) + len(c_inds))

df_num = _df3.iloc[:, n_inds]
df_cat = _df3.iloc[:, c_inds]

# music genre dummies
songs = _df3.iloc[:, 42].str.get_dummies(sep=', ').reset_index(drop=True)
df_cat2 = pd.get_dummies(df_cat)

# convert to numeric
# convert df_cat2 to numeric
conv_to_numeric(df_cat2, list(range(len(df_cat2.columns))))

X = pd.concat([df_num, df_cat2, songs], axis=1)
length = len(df_cat2.columns) + len(songs.columns) + len(df_num.columns)
print()
print("No. of columns after dummification:", len(X.columns))
```

```
No. of columns containing non-numerical data: 15
No. of columns prior to dummification: 44
No. of columns that must be changed to numerical: 120
No. of columns automatically changed to numbers, 1st pass: 120

Remaining columns with values that must be changed to numbers:
All values numerical already.

No. of columns after dummification: 170
```

`y_temp` values are converted to 1 for "yes", 0 for "no", and 2 for "it's complicated" and saved to `y`. We can do this because in this case, the numbers are just placeholders for the category names. They do not have mathematical interaction with the classification model except being the target classes.

In [17]:
```
y_ = []
for i in y_temp:
    if i == 'yes':
        y_.append(1)
    elif i == "no":
        y_.append(0)
    else:
        y_.append(2)

y = pd.Series(y_, name='Target')
print("y contains the three classes and {} for observations".format(len(y)))
```

```
y contains the three classes and 35 for observations
```

Finally, we make another set of dataframes, this time, dataframes of "undummified" categories plus the targets added to the last column. The same was done for numerical categories. These will be used for data exploration below.

In [18]:
```
df_cat_targ = pd.concat([df_cat, y], axis=1) # get categories with targets
df_num_targ = pd.concat([df_num, y], axis=1) # get num categories with targets
```

Exploring the Data

15 respondents are not in a relationship, 19 are in a relationship, and 1 is in a complicated relationship.

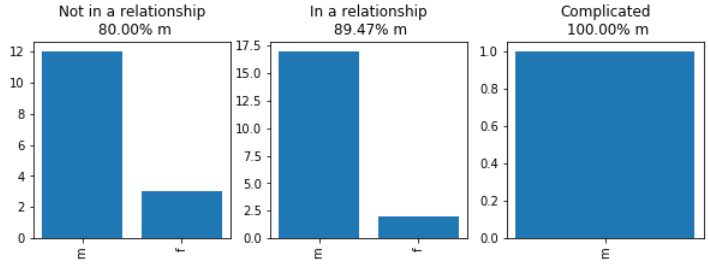
In [19]:
```
print("No. of respondents not in a relationship:",np.sum(y == 0))
print("No. of respondents in a relationship:",np.sum(y == 1))
print("No. of respondents not in a complicated relationship:",np.sum(y == 2))
```

```
No. of respondents not in a relationship: 15
No. of respondents in a relationship: 19
No. of respondents not in a complicated relationship: 1
```

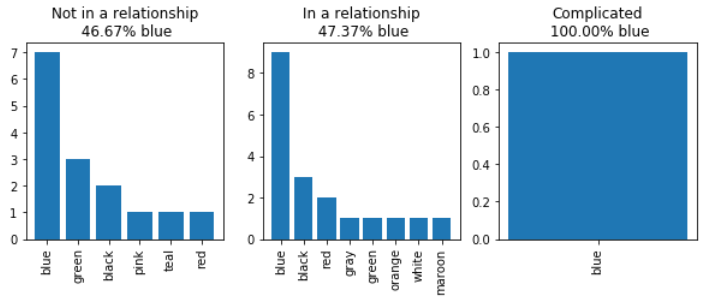
The frequencies of answers for categorical questions are shown in the bar charts below. The charts are divided according to relationship status. Just by looking at the bar charts, we get to see that the kinds of answers are more or less the same regardless of the relationship status.

In [20]:
```
big_list = []

titles = ['Not in a relationship', 'In a relationship', 'Complicated']

for n in range(len(df_cat.columns)):

    small_list = [df_cat.columns[n]]

    plt.figure(figsize=(10, 3))
    for i in range(3):
        plt.subplot(1, 3, i + 1)
        keys = list(
            dict(df_cat_targ.iloc[:, n][df_cat_targ.Target == i].value_counts()).keys())
        vals = list(
            dict(df_cat_targ.iloc[:, n][df_cat_targ.Target == i].value_counts()).values())

        inds = np.argsort(vals)
        sorted_keys = np.array(keys)[inds]

        plt.bar(sorted(keys), sorted(vals)[::-1])
        plt.xticks(np.arange(len(keys)),
                   sorted_keys[::-1], rotation='vertical')

        m = max(vals) / np.sum(vals) * 100  # proportion of max value
        v = keys[vals.index(max(vals))]  # most numerous variable

        plt.title(titles[i] + "\n {0:.2f}% ".format(m) + str(v))
        small_list.append("{0:.2f}% ".format(m) + str(v))

    big_list.append(small_list)

    t = df_cat_targ.columns[n]  # title/column
    print(t)
    print('=============')

    plt.show()
```

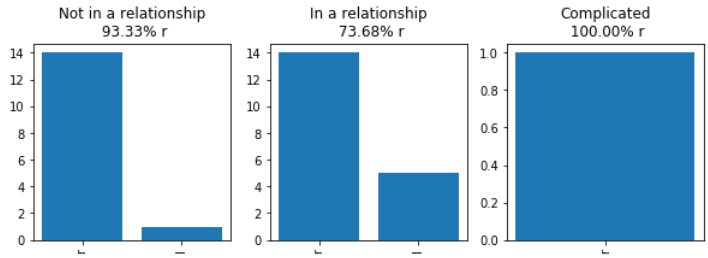
Male or female? (pls type M or F)
==========



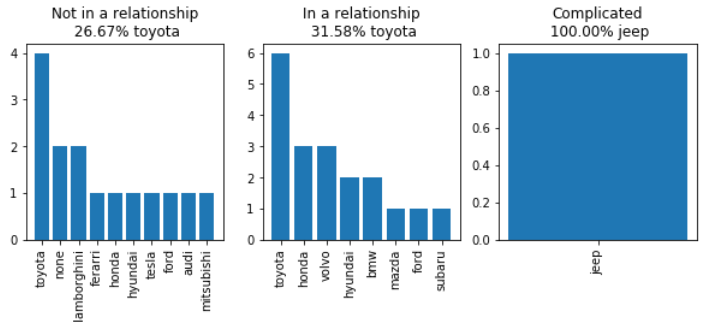
What is your favorite color?
==========



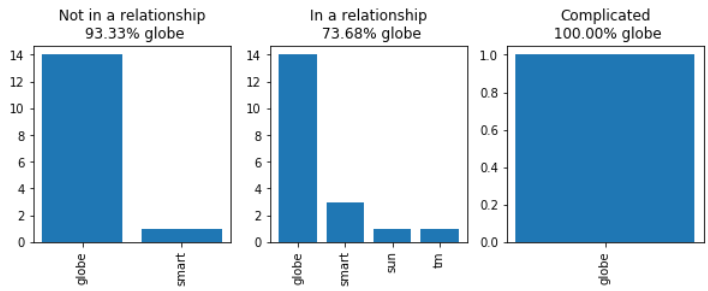
Are you predominantly right handed or left handed? (pls type R or L)
==========



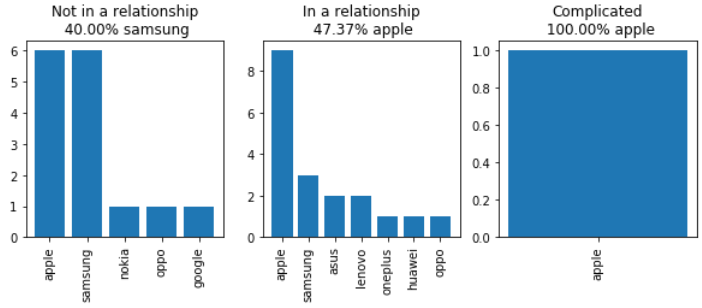
What is your favorite car brand? Toyota, Honda, Hyundai, BMW, Mercedes, etc.
==========



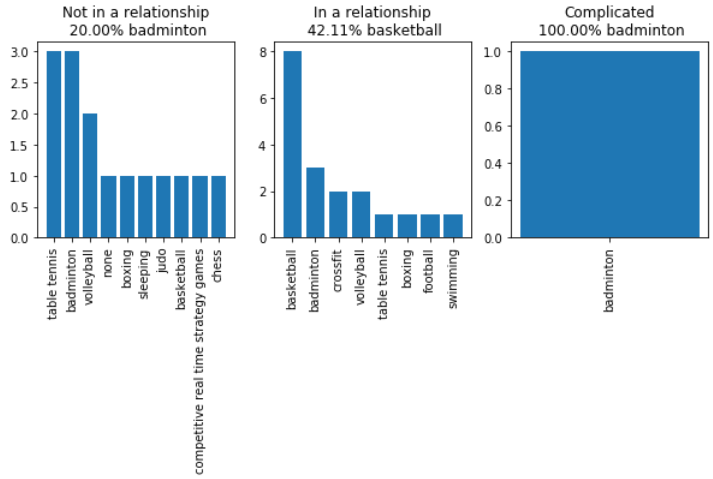
What phone network do you use? Globe, Smart, Sun, etc.
==========



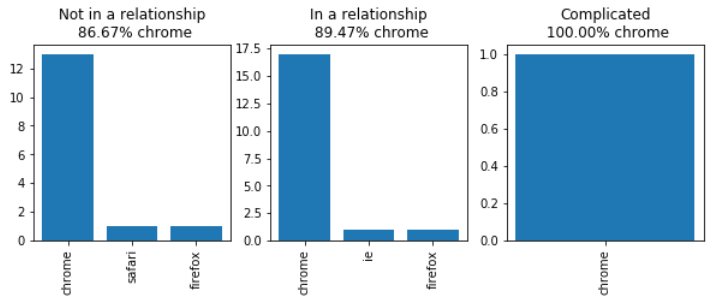
What phone brand do you use? Samsung, Apple, Vivo, Huawei, Sony, etc.
==========



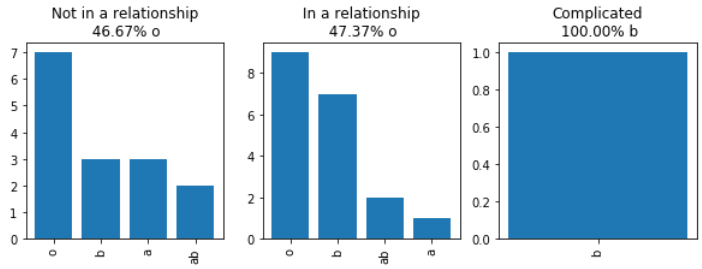
What is your favorite sport? Basketball, football, volleyball, tennis, etc.
==========



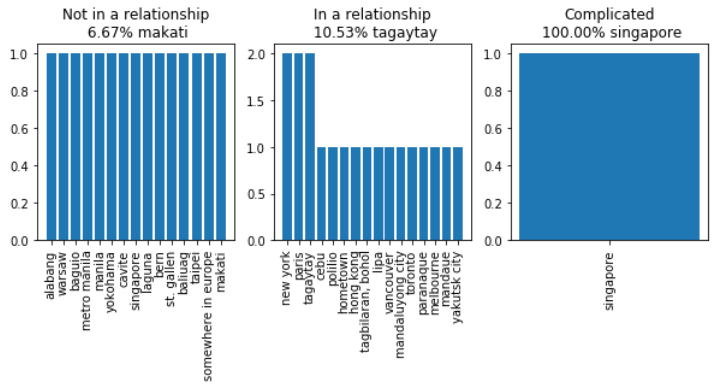
What browser do you prefer? Chrome, IE, Firefox, Safari
==========



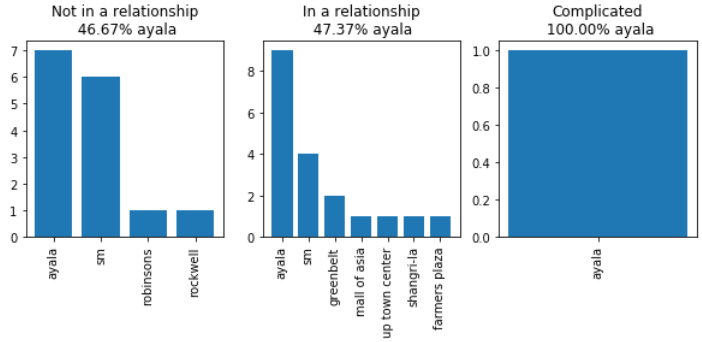
What is your blood type? O A B AB
==========



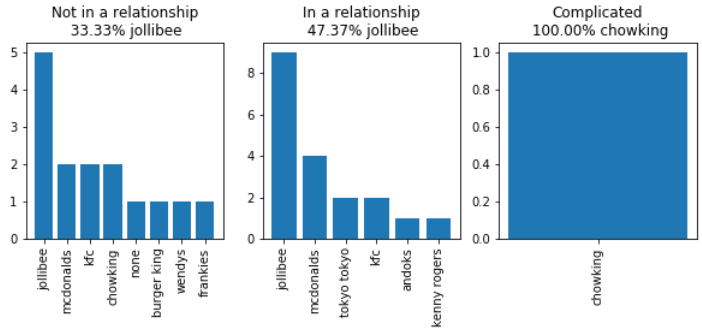
Where is your ideal city to live in?
==========



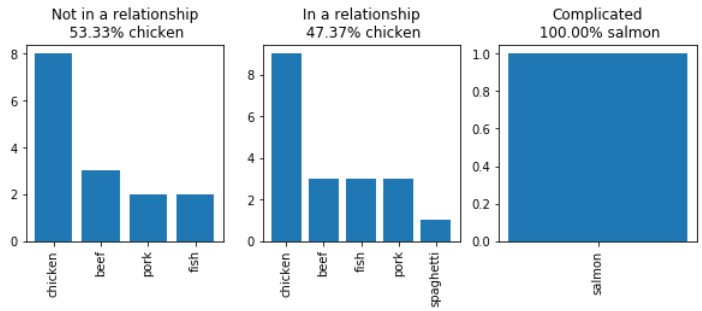
```
What is your favorite mall? SM, Ayala, Robinsons, Rustan's, etc.
=============
```




```
What is your favorite fastfood place?
=============
```




```
What is your favorite meat to eat? Chicken, pork, beef, fish, etc.
=============
```



The top answers for categorical questions split based on relationship status are summarized in the table below. It seems that answers are consistant between those not in a relationship and those in a relationship. Note that there's only one respondent for the column `Complicated`, which is why all values are 100%.

In [21]:
```
print("Most Frequent Responses")
print("======================")
cols = ['Question','Not in a relationship', 'In a relationship', 'Complicated']
display(pd.DataFrame(big_list, columns=cols))
```

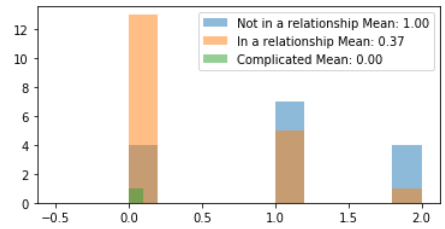
```
Most Frequent Responses
======================
```

	Question	Not in a relationship	In a relationship	Complicated
0	Male or female? (pls type M or F)	80.00% m	89.47% m	100.00% m
1	What is your favorite color?	46.67% blue	47.37% blue	100.00% blue
2	Are you predominantly right handed or left handed? (pls type R or L)	93.33% r	73.68% r	100.00% r
3	What is your favorite car brand? Toyota, Honda, Hyundai, BMW, Mercedes, etc.	26.67% toyota	31.58% toyota	100.00% jeep
4	What phone network do you use? Globe, Smart, Sun, etc.	93.33% globe	73.68% globe	100.00% globe
5	What phone brand do you use? Samsung, Apple, Vivo, Huawei, Sony, etc.	40.00% samsung	47.37% apple	100.00% apple
6	What is your favorite sport? Basketball, football, volleyball, tennis, etc.	20.00% badminton	42.11% basketball	100.00% badminton
7	What browser do you prefer? Chrome, IE, Firefox, Safari	86.67% chrome	89.47% chrome	100.00% chrome
8	What is your blood type? O A B AB	46.67% o	47.37% o	100.00% b
9	Where is your ideal city to live in?	6.67% makati	10.53% tagaytay	100.00% singapore
10	What is your favorite mall? SM, Ayala, Robinsons, Rustan's, etc.	46.67% ayala	47.37% ayala	100.00% ayala
11	What is your favorite fastfood place?	33.33% jollibee	47.37% jollibee	100.00% chowking
12	What is your favorite meat to eat? Chicken, pork, beef, fish, etc.	53.33% chicken	47.37% chicken	100.00% salmon

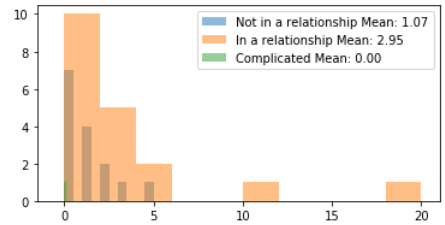
The histograms of numerical answers from students from the three relationship types seem to overlap for most of the questions, except for `How many times in a month do you go out with your s.o.?` which clearly shows a segregation between the relationship types. Classes are similarly segregated for `How many partners have you had?`.

In [22]:
```
big_list2 = []

titles = ['Not in a relationship', 'In a relationship', 'Complicated']

for n in range(len(df_num.columns)):

    small_list = [df_num.columns[n]]

    plt.figure(figsize=(6, 3))
    for i in range(3):
        plt.gca()
        vals = list(df_num_targ.iloc[:, n][df_num_targ.Target == i])

        m = np.mean(vals)  # mean value

        plt.hist(vals, alpha=0.5, label=titles[i]+" Mean: {0:.2f}".format((m)))
        plt.legend()

        small_list.append("{0:.2f}".format(m))

    big_list2.append(small_list)

    t = df_num_targ.columns[n]  # title/column
    print(t)
    print('=============')

    plt.show()
```

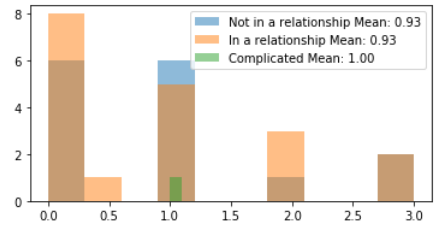
Number of movies you watched from cinema in the last two weeks
==========



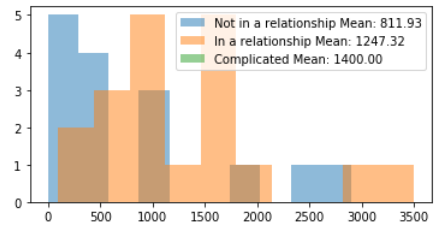
Number of movies you watched from home in the last two weeks
==========



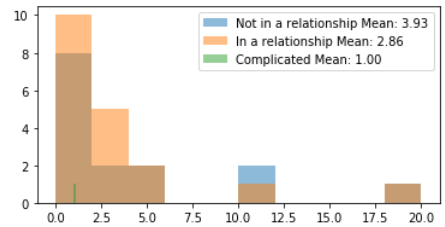
Average Cups of coffee consumed per day
==========



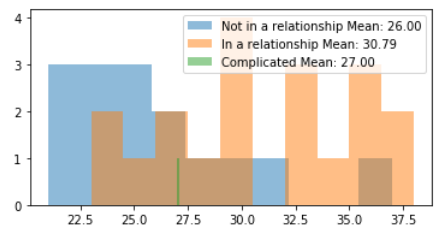
Average amount of bills paid for mobile used
==========



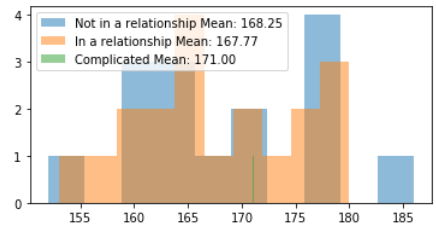
Average number of posts in social media per week
==========



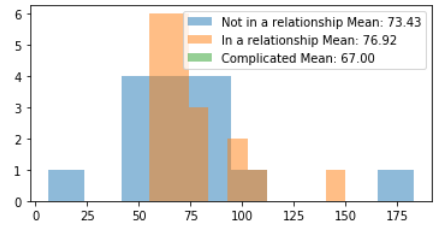
How old are you?
==========



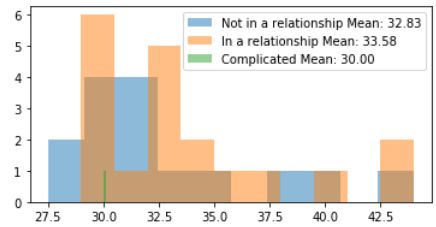
What is your height in cm?
==========



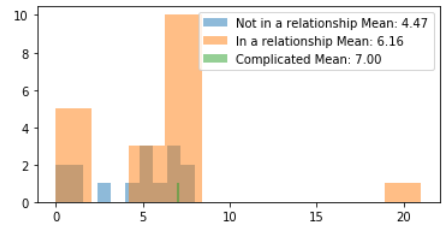
What is your weight in kg?
==========



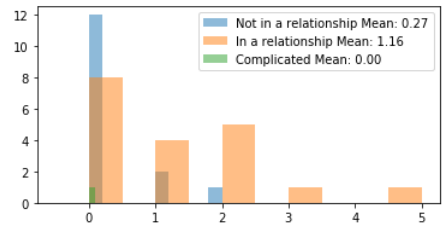
What is your waist size in inches?
==========



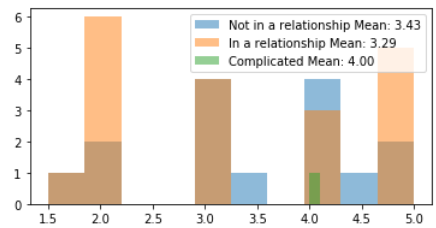
What is your favorite number, from 0 to 9?
==========



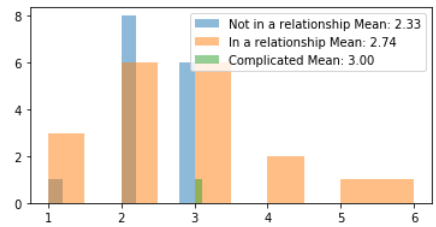
How many cars have you owned in your life?
==========



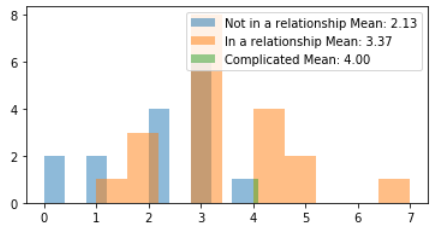
How many years do you use your phone before replacing it?
==========



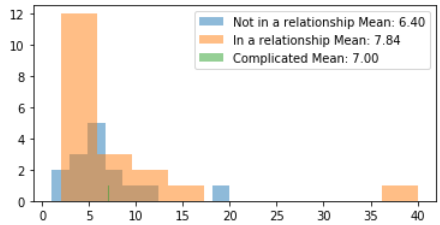
How many cities have you lived in? (Must have lived more than 1 year)
==========



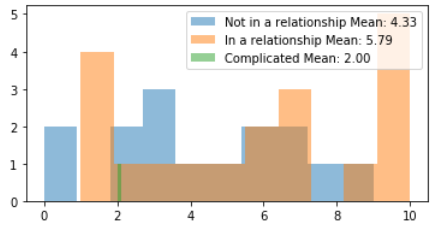
How many siblings are you in the family? (not counting half siblings)
==========



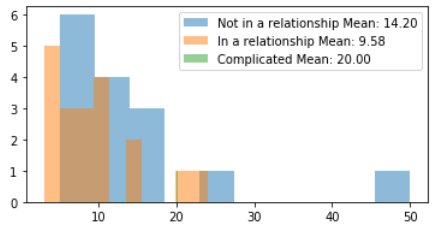
How many functioning shoes do you own currently?
==========



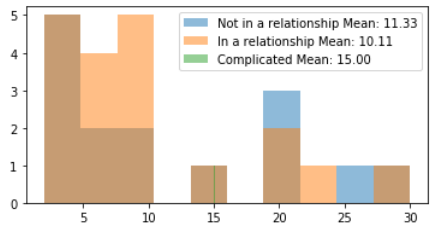
How many countries have you visited?
==========



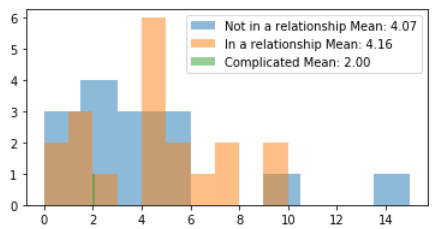
How many provinces in the Philippines have you visited?
==========



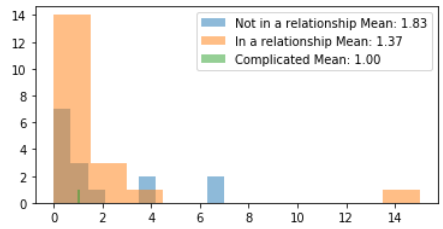
How many times in a month do you go to the mall?
==========



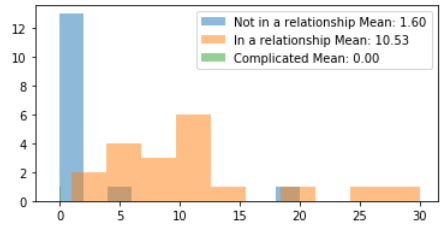
How many times in a month do you eat in Jollibee?
==========



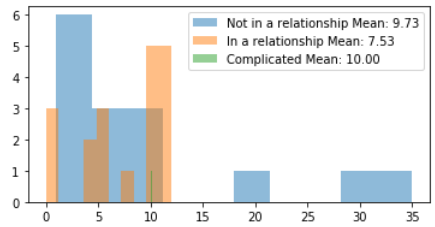
How many times in a week do you cook at home?
==========



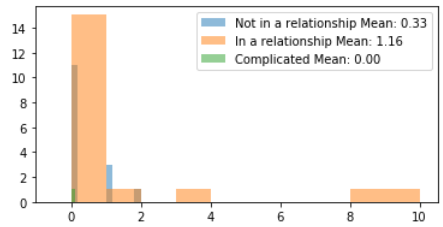
How many times in a month do you go out with your s.o.?
==========



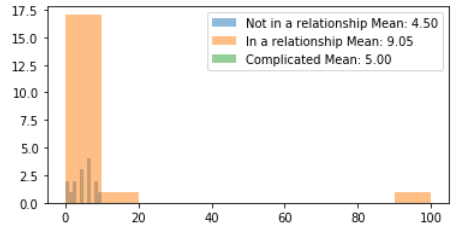
How many movies do you see inside a theater in a year?
==========



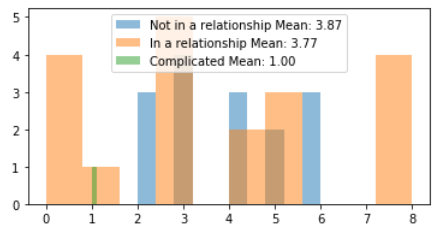
How many years in your life have you been a smoker?
==========



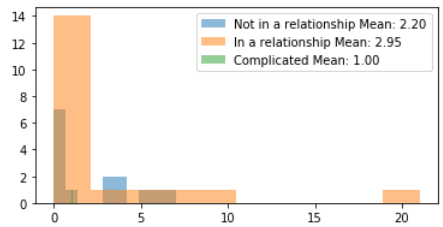
How many beers can you drink in one night?
==========



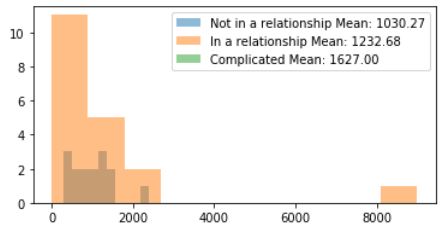
How many pizzas can you eat in one sitting?
==========



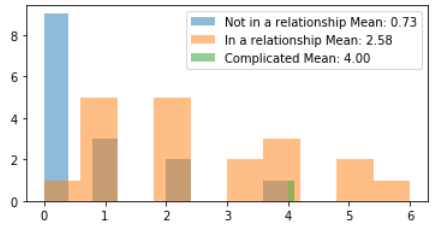
How many times do you buy coffee in a week?
==========



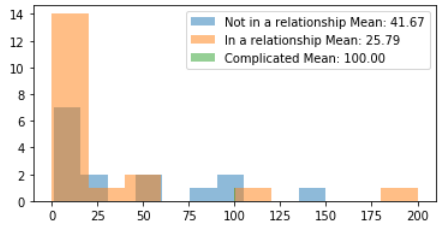
```
How many Facebook friends do you have? (In hundreds, e.g. 1100)
=============
```



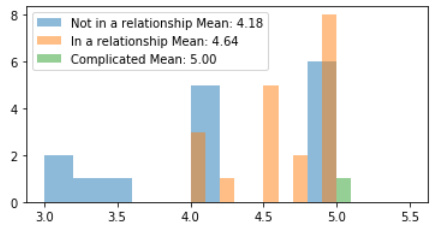

```
How many partners have you had?
=============
```



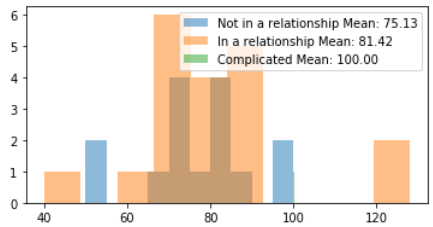

```
How many songs do you know the lyrics to?
=============
```




```
What do you think your grade in this class will be?
=============
```




```
What age do you think you will live to?
=============
```



The table below summarizes the numerical responses for each question and relationship status. We also see that on the average, those not in a relationship watch movies in theatres more often than those in relationships. However, those in relationships tend to watch movies at home. Those not in a relationship spend the least for mobile bills.

And as observed in the histograms above, the number of times the student went out with his/her special someone as well as the number of partners the student had, are much higher for those in a relationship.

In [23]:
```
cols = ['Question','Not in a relationship', 'In a relationship', 'Complicated']
display(pd.DataFrame(big_list2, columns=cols))
```

	Question	Not in a relationship	In a relationship	Complicated
0	Number of movies you watched from cinema in the last two weeks	1.00	0.37	0.00
1	Number of movies you watched from home in the last two weeks	1.07	2.95	0.00
2	Average Cups of coffee consumed per day	0.93	0.93	1.00
3	Average amount of bills paid for mobile used	811.93	1247.32	1400.00
4	Average number of posts in social media per week	3.93	2.86	1.00
5	How old are you?	26.00	30.79	27.00
6	What is your height in cm?	168.25	167.77	171.00
7	What is your weight in kg?	73.43	76.92	67.00
8	What is your waist size in inches?	32.83	33.58	30.00
9	What is your favorite number, from 0 to 9?	4.47	6.16	7.00
10	How many cars have you owned in your life?	0.27	1.16	0.00
11	How many years do you use your phone before replacing it?	3.43	3.29	4.00
12	How many cities have you lived in? (Must have lived more than 1 year)	2.33	2.74	3.00
13	How many siblings are you in the family? (not counting half siblings)	2.13	3.37	4.00
14	How many functioning shoes do you own currently?	6.40	7.84	7.00
15	How many countries have you visited?	4.33	5.79	2.00
16	How many provinces in the Philippines have you visited?	14.20	9.58	20.00
17	How many times in a month do you go to the mall?	11.33	10.11	15.00
18	How many times in a month do you eat in Jollibee?	4.07	4.16	2.00
19	How many times in a week do you cook at home?	1.83	1.37	1.00
20	How many times in a month do you go out with your s.o.?	1.60	10.53	0.00
21	How many movies do you see inside a theater in a year?	9.73	7.53	10.00
22	How many years in your life have you been a smoker?	0.33	1.16	0.00
23	How many beers can you drink in one night?	4.50	9.05	5.00
24	How many pizzas can you eat in one sitting?	3.87	3.77	1.00
25	How many times do you buy coffee in a week?	2.20	2.95	1.00
26	How many Facebook friends do you have? (In hundreds, e.g. 1100)	1030.27	1232.68	1627.00
27	How many partners have you had?	0.73	2.58	4.00
28	How many songs do you know the lyrics to?	41.67	25.79	100.00
29	What do you think your grade in this class will be?	4.18	4.64	5.00
30	What age do you think you will live to?	75.13	81.42	100.00

Models

Proportional Chance Criterion

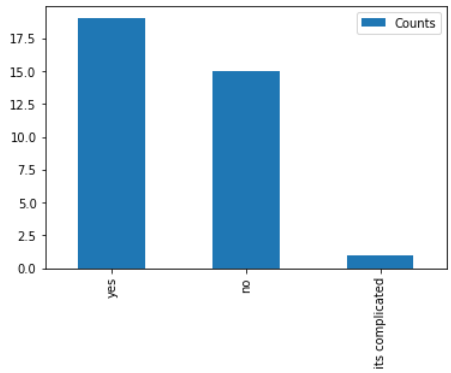
The Proportional Chance Criterion (PCC) measures the chance of correctly classifying a datapoint based on chance alone. As a rule of thumb, to say that our model works, we need to exceed prediction accuracy of 1.25 x PCC. In this case, we need to exceed 60% accuracy.

In [24]:
```
state_counts = Counter(y_temp)
df_state = pd.DataFrame.from_dict(state_counts, orient='index')
df_state.columns = ['Counts']
df_state.plot(kind='bar')
print("Population per class:")
display(df_state)
num = (df_state['Counts'] / df_state['Counts'].sum())**2
print(
    "1.25 * Proportion Chance Criterion: {0:.2f}%".format(1.25 * 100 * num.sum()))
```

```
Population per class:
```

	Counts
yes	19
no	15
its complicated	1

```
1.25 * Proportion Chance Criterion: 59.90%
```



k Nearest Neighbors Classifier

Here, we run a k-Nearest Neighbors algorithm to automatically classify whether a person is in a relationship or not.

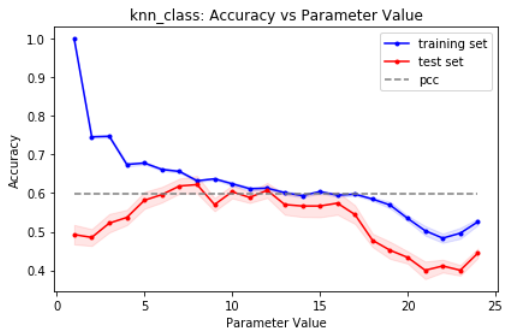
kNN is an algorithm that we can use to classify a new observation based on the classifications of historical observations. Loosely speaking, new observations are classified based on how closely they resemble the features of historical observations.

The important parameter of kNN is k , which refers to the number of nearest neighbors or number of training datapoints with features closest to the features of the new datapoint being classified. The more nearest neighbors considered (larger k), the more the model considers the training data in general. The fewer nearest neighbors considered (smaller k), the more the model considers individual datapoints. Too high k and the model risks underfitting. Too low k and the model risks overfitting.

The chart below shows the accuracy vs parameter value (nearest neighbors, k) on the training set (blue line) and test set (red line). The red and blue lines show the average accuracies from 30 iterations (different sampling of training and testing set from the original dataset). The proportional chance criterion is plotted as a dashed horizontal gray line. The transparent colored areas around the lines show the extent of one standard deviation from the average.

We are interested in the model's performance or accuracy on the test set (red line). We observe that running all features on kNN gives a classification accuracy of only 62% using an optimal parameter of 8 nearest neighbors. Although this exceeds the PCC, further analysis will show that reducing the number of features will improve accuracy.

In [25]:
```
ml_class(X, y, ml_type='knn_class', show_PCC=True,
         param_range=range(1, 25), seed_settings=range(0, 30),
         plot=True, report=True);
```




```
Report:
=======
Max average accuracy: 0.6222
Var of accuracy at optimal parameter: 0.0227
Optimal parameter: 8.0000
1.25 x PCC: 0.5990
Total iterations: 30
```

Unsurprisingly, when the model is run one feature at a time, we find the number one single predictor is how many times a month the person goes out with his/her loved one, predicting at 90% accuracy. The second predictor is similarly unsurprising: the number of partners the person has. The third highest individual predictor is the number of movies the person sees in a year. It might be possible that those in relationships go to movies during their dates with their special other.

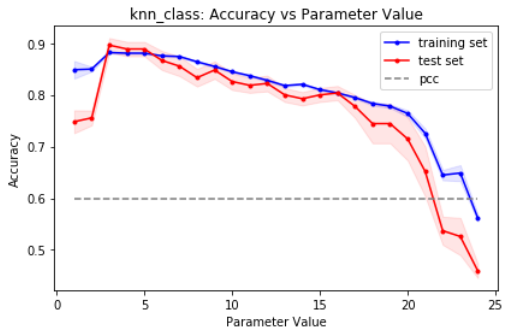
In [26]:
```
acc = []
params = range(1, 10)

for i in range(len(X.columns)):
    x = pd.DataFrame(X.iloc[:, i])
    a, p = ml_class(x, y, ml_type='knn_class', show_PCC=False,
                    param_range=params, seed_settings=range(0, 30),
                    plot=False, report=False)
    acc.append(a)

inds = np.argsort(acc)[::-1]
sorted_acc = np.array(acc)[inds]
cols = np.array(X.columns)[inds]
```

In [27]:
```
g = pd.DataFrame(sorted_acc, cols)
g.columns = ['Accuracy']
g.head(5)
```

Out[27]:

	Accuracy
How many times in a month do you go out with your s.o.?	0.8963
How many partners have you had?	0.7259
How many movies do you see inside a theater in a year?	0.7148
Number of movies you watched from cinema in the last two weeks	0.6889
Average amount of bills paid for mobile used	0.6889

In [28]:
```
ml_class(X[cols[:1]], y, ml_type='knn_class', show_PCC=True,
         param_range=range(1, 25), seed_settings=range(0, 30),
         plot=True, report=True);
```




```
Report:
=======
Max average accuracy: 0.8963
Var of accuracy at optimal parameter: 0.0139
Optimal parameter: 3.0000
1.25 x PCC: 0.5990
Total iterations: 30
```

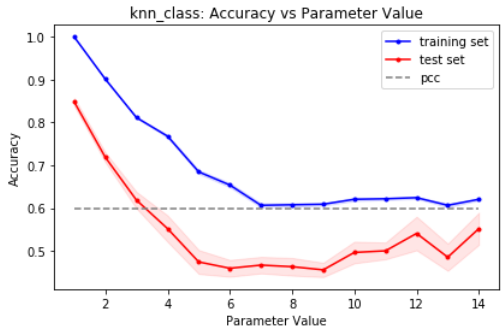
As seen in the table and chart above, rerunning the model on the top feature `How many times in a month do you go out with your s.o.` predicts the target with around 90% accuracy. However, this is expected since you can only go out with your partner if you have a partner. The more interesting question is whether or not we can predict a students' relationship status based on the other seemingly unrelated features. We also exclude `How many partners have you had?`

Using trial and error on the top 50 features identified above (only five shown), we can use 11 features and get 85% accuracy at 1 nearest neighbor.

In [29]:
```
params = range(1, 25)
lst = []
accs = [0]
p_opts = [0]

for i in range(2, 50):
    lst2 = lst[:]
    lst2.append(i)
    acc, p_opt = ml_class(X[cols[lst2]], y, ml_type='knn_class', show_PCC=False,
                          param_range=params, seed_settings=range(0, 30),
                          plot=False, report=False)
    if acc > accs[-1]:
        accs.append(acc)
        p_opts.append(p_opt)
        lst.append(i)
```

In [30]:
```
print(lst)
acc_knn, op_knn = ml_class(X[cols[lst]], y, ml_type='knn_class',
                           show_PCC=True, param_range=range(1, 15),
                           seed_settings=range(0, 30),
                           plot=True, report=True)
```

```
[2, 3, 8, 10, 11, 13, 22, 24, 30, 33, 38]
```




```
Report:
=======
Max average accuracy: 0.8481
Var of accuracy at optimal parameter: 0.0095
Optimal parameter: 1.0000
1.25 x PCC: 0.5990
Total iterations: 30
```

Below are the 11 features used:

In [31]:
```
print("Features that resulted in 83% accuracy:\n",
      [cols[lst]])
```

```
Features that resulted in 83% accuracy:
 [array(['How many movies do you see inside a theater in a year?',
       'Number of movies you watched from cinema in the last two weeks',
       'What phone brand do you use? Samsung, Apple, Vivo, Huawei, Sony, etc._samsung',
       'What is your favorite sport? Basketball, football, volleyball, tennis, etc._basketball',
       'What is your favorite number, from 0 to 9?',
       'How many siblings are you in the family? (not counting half siblings)',
       'What is your favorite car brand? Toyota, Honda, Hyundai, BMW, Mercedes, etc._none',
       'What is your favorite fastfood place?_chowking',
       'What is your favorite color?_green',
       'What is your favorite car brand? Toyota, Honda, Hyundai, BMW, Mercedes, etc._lamborghini',
       'What is your favorite car brand? Toyota, Honda, Hyundai, BMW, Mercedes, etc._jeep'],
      dtype=object)]
```

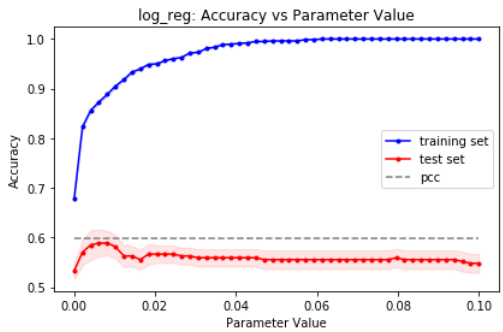
Logistic Regression

Logistic Regression is another classification algorithm (even though it's called regression) derived from linear regression, but instead of calculating a continuous outcome, it calculates the most probable categorical outcome. To illustrate this, say we want to predict the score of an athlete in a competition, given the hours of practicing, then linear regression may be used. But if we want to calculate the odds that the athlete will win the competition, then we can use logistic regression.

The important parameter of logistic regression, C, can be looked at as like a "regularization" or "generalization" factor. The higher the value of C, the more the classifier gives weight to individual datapoints, while the lower its value, the more the classifier gives weight to the training dataset as a whole. C with too high a value results in overfitting the training data, which reduces generalizability on new datapoints. While C with too low a value results in underfitting, and results in too low accuracies on either the training data or test data.

Applying logistic regression using all features yields an accuracy of around 60%. Reducing the number of features will significantly improve accuracy as will be shown below.

In [32]:
```
params = np.linspace(0.0001, 0.1, 50)
a, p_optimal = ml_class(X, y, ml_type='log_reg', show_PCC=True,
                        param_range=params, seed_settings=range(0, 30),
                        plot=True, report=True, penalty='l2')
```




```
Report:
=======
Max average accuracy: 0.5889
Var of accuracy at optimal parameter: 0.0273
Optimal parameter: 0.0062
1.25 x PCC: 0.5990
Total iterations: 30
```

After running Logistic regression one feature at a time, the top predictor is also the number of times per month that the person goes out with his/her special other. Using this feature alone will yield 93% classification accuracy at C = 0.0273.

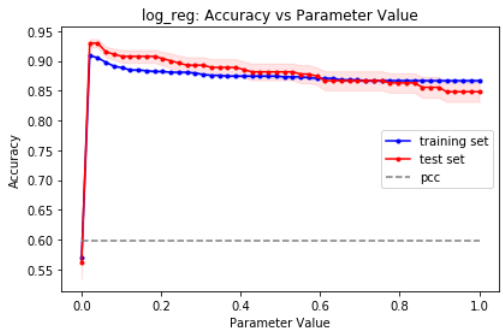
In [33]:
```
acc = []
params = np.linspace(0.0001, 1, 30)

for i in range(len(X.columns)):
    x = pd.DataFrame(X.iloc[:, i])
    a, p = ml_class(x, y, ml_type='log_reg', show_PCC=False,
                    param_range=params, seed_settings=range(0, 30),
                    plot=False, report=False, penalty='l1')
    acc.append(a)

inds = np.argsort(acc)[::-1]
sorted_acc = np.array(acc)[inds]
cols = np.array(X.columns)[inds]
```

In [34]:
```
g = pd.DataFrame(sorted_acc, cols)
g.columns = ['Accuracy']
g.head(5)
```

Out[34]:

	Accuracy
How many times in a month do you go out with your s.o.?	0.9296
How many partners have you had?	0.7778
Number of movies you watched from cinema in the last two weeks	0.6704
How many siblings are you in the family? (not counting half siblings)	0.6444
How many cars have you owned in your life?	0.6370

In [35]:
```
params = np.linspace(0.0001, 1, 50)
ml_class(X[cols[0:1]], y, ml_type='log_reg', show_PCC=True,
         param_range=params, seed_settings=range(0, 30),
         plot=True, report=True, penalty = 'l2');
```




```
Report:
=======
Max average accuracy: 0.9296
Var of accuracy at optimal parameter: 0.0070
Optimal parameter: 0.0205
1.25 x PCC: 0.5990
Total iterations: 30
```

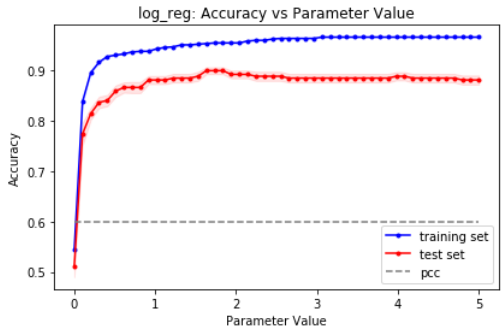
Similar to kNN, rerunning logistic regression on the top feature `How many times in a month do you go out with your s.o.` predicts the target with around 93% accuracy. Can we find the other features that will still classify a student's relationship status excluding this feature and the feature `How many partners have you had?`

Using trial and error on the top 50 features identified above (only five shown), we find that we can use 9 features and get 90% accuracy at C = 1.6327.

In [36]:
```
params = np.linspace(0.0001, 5, 50)
lst = []
accs = [0]
p_opts = [0]

for i in range(2, 50):
    lst2 = lst[:]
    lst2.append(i)
    acc, p_opt = ml_class(X[cols[lst2]], y, ml_type='log_reg', show_PCC=True,
                          param_range=params, seed_settings=range(0, 30),
                          plot=False, report=False, penalty='l2')
    if acc > accs[-1]:
        accs.append(acc)
        p_opts.append(p_opt)
        lst.append(i)
```

In [37]:
```
print(lst)
acc_logreg, op_logreg = ml_class(X[cols[lst]], y, ml_type='log_reg',
                                 show_PCC=True,
                                 param_range=params, seed_settings=range(
                                     0, 30),
                                 plot=True, report=True, penalty='l2')
```

```
[2, 3, 4, 8, 15, 25, 28, 39, 43]
```




```
Report:
=======
Max average accuracy: 0.9
Var of accuracy at optimal parameter: 0.0069
Optimal parameter: 1.6327
1.25 x PCC: 0.5990
Total iterations: 30
```

The features used are:

In [38]:
```
print("Top Features:\n", [cols[lst]])
```

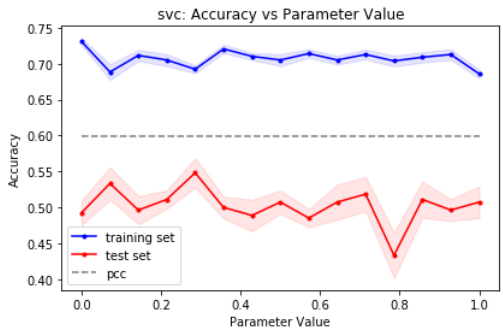
```
Top Features:
 [array(['Number of movies you watched from cinema in the last two weeks',
       'How many siblings are you in the family? (not counting half siblings)',
       'How many cars have you owned in your life?',
       'What is your favorite number, from 0 to 9?',
       'What is your waist size in inches?',
       'What phone brand do you use? Samsung, Apple, Vivo, Huawei, Sony, etc._samsung',
       'What is your favorite car brand? Toyota, Honda, Hyundai, BMW, Mercedes, etc._mitsubishi',
       'What phone brand do you use? Samsung, Apple, Vivo, Huawei, Sony, etc._asus',
       'What phone brand do you use? Samsung, Apple, Vivo, Huawei, Sony, etc._nokia'],
      dtype=object)]
```

Linear SVC

Linear Support Vector Classifier (SVC) is another classification algorithm that works by finding the lines, planes, or hyperplanes (in the case beyond three features or dimensions) that divide the observations into two or more classes. In this case, the classes are whether or not the person is in a relationship. The important parameter C for Linear SVC works like in logistic regression.

Applying Linear SVC on all features yields 55% accuracy, below the minimum target of 60%. Again, reducing the features to one will significantly improve accuracy as shown below.

In [39]:
```
params = np.linspace(0.0001, 1, 15)
ml_class(X, y, ml_type='svc', show_PCC=True,
         param_range=params, seed_settings=range(0, 30),
         plot=True, report=True, penalty='l2');
```




```
Report:
=======
Max average accuracy: 0.5481
Var of accuracy at optimal parameter: 0.0205
Optimal parameter: 0.2858
1.25 x PCC: 0.5990
Total iterations: 30
```

After running SVC on each feature, the sole feature that maximizes accuracy is the number of times per month the person goes out with the special other, same as above. Using this feature alone, the maximum average accuracy obtained is around 91%.

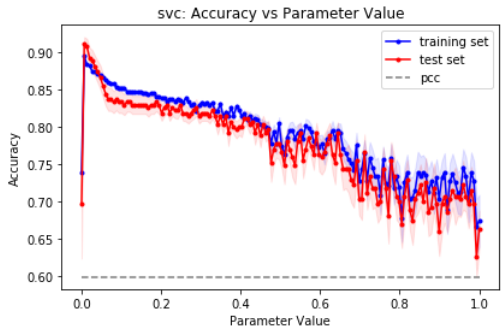
In [40]:
```
acc = []
params = np.linspace(0.0001, 1, 50)

for i in range(len(X.columns)):
    x = pd.DataFrame(X.iloc[:, i])
    a, p = ml_class(x, y, ml_type='svc', show_PCC=False,
                    param_range=params, seed_settings=range(0, 30),
                    plot=False, report=False)
    acc.append(a)

inds = np.argsort(acc)[::-1]
sorted_acc = np.array(acc)[inds]
cols = np.array(X.columns)[inds]
```

In [41]:
```
g = pd.DataFrame(sorted_acc, cols)[:5]
g.columns = ['Accuracy']
g.head(5)
```

Out[41]:

	Accuracy
How many times in a month do you go out with your s.o.?	0.8926
How many partners have you had?	0.7778
Number of movies you watched from cinema in the last two weeks	0.6704
How many siblings are you in the family? (not counting half siblings)	0.6519
How old are you?	0.6407

In [42]:
```
params = np.linspace(0.0001, 1, 150)
ml_class(X[cols[:1]], y, ml_type='svc', show_PCC=True,
         param_range=params, seed_settings=range(0, 30),
         plot=True, report=True);
```




```
Report:
=======
Max average accuracy: 0.9111
Var of accuracy at optimal parameter: 0.0094
Optimal parameter: 0.0068
1.25 x PCC: 0.5990
Total iterations: 30
```

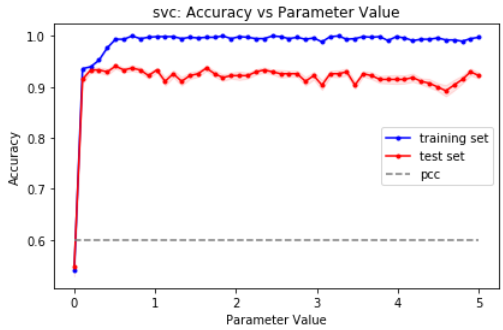
Similar to logistic regression, rerunning logistic regression on the top feature `How many times in a month do you go out with your s.o.` predicts the target with around 91% accuracy. Again we find other features that will still classify a student's relationship status excluding this feature and the feature `How many partners have you had?`

Using trial and error on the top 50 features identified above (only five shown), we find that we can use 13 features and get 94% accuracy at C = 0.5103.

In [43]:
```
params = np.linspace(0.0001, 5, 50)
lst = []
accs = [0]
p_opts = [0]

for i in range(2, 50):
    lst2 = lst[:]
    lst2.append(i)
    acc, p_opt = ml_class(X[cols[lst2]], y, ml_type='svc', show_PCC=True,
                          param_range=params, seed_settings=range(0, 30),
                          plot=False, report=False)
    if acc > accs[-1]:
        accs.append(acc)
        p_opts.append(p_opt)
        lst.append(i)
```

In [44]:
```
print(lst)
acc_svc, op_svc = ml_class(X[cols[lst]], y, ml_type='svc', show_PCC=True,
                           param_range=params, seed_settings=range(0, 30),
                           plot=True, report=True)
```

```
[2, 3, 4, 6, 7, 10, 12, 15, 16, 20, 21, 32, 42]
```




```
Report:
=======
Max average accuracy: 0.9407
Var of accuracy at optimal parameter: 0.0047
Optimal parameter: 0.5103
1.25 x PCC: 0.5990
Total iterations: 30
```

The features used are:

In [45]:
```
print("Top Features:\n", [cols[lst]])
```

```
Top Features:
 [array(['Number of movies you watched from cinema in the last two weeks',
       'How many siblings are you in the family? (not counting half siblings)',
       'How old are you?',
       'What phone brand do you use? Samsung, Apple, Vivo, Huawei, Sony, etc._samsung',
       'How many cars have you owned in your life?', 'indie',
       'What is your favorite sport? Basketball, football, volleyball, tennis, etc._table tennis',
       'What is your favorite fastfood place?_chowking',
       'What is your favorite color?_green',
       'What is your favorite mall? SM, Ayala, Robinsons, Rustan's, etc._sm',
       'What is your favorite number, from 0 to 9?',
       'What is your favorite mall? SM, Ayala, Robinsons, Rustan's, etc._greenbelt',
       'What is your favorite sport? Basketball, football, volleyball, tennis, etc._none'],
      dtype=object)]
```

The results of the runs are summarized in the table below.

In [46]:
```
models = pd.Series(['kNN', 'Logistic Regression', 'SVC'], name='models')
accs = pd.Series([acc_knn, acc_logreg, acc_svc], name='Accuracies')
params = pd.Series([op_knn, op_logreg, op_svc], name='Optimal Parameter Value')
pd.concat([models, accs, params], axis=1)
```

Out[46]:

	models	Accuracies	Optimal Parameter Value
0	kNN	0.8481	1.000000
1	Logistic Regression	0.9000	1.632720
2	SVC	0.9407	0.510294

Results

Using all features in the dataset yielded low accuracies for all three models used. Reducing the number of features improved the model results.

As expected, the single best predictor for whether or not a person is in a relationship, not in a relationship, or in a complicated relationship is `How many times in a month do you go out with your s.o`. The second best predictor is `How many partners have you had?`. It might be possible that those students in a relationship had strong affinity to be in a relationship. The sole student in a complicated relationship have had four partners. The third predictor is `How many movies do you see inside a theater in a year?` (kNN) or `Number of movies you watched from cinema in the last two weeks` (logistic regression and Linear SVC). Looking back at the histograms, we see that in general, those not in a relationship watch movies in theatres more frequently than those in relationships. Might it be possible that these students actually watch movies more because they're in the dating phase?

The top three features that give the highest accuracy when taken individually are shown in the table below.

kNN Accuracy	**Logistic Regression Accuracy**	**Linear SVC Accuracy**
How many times in a month do you go out with your s.o.?	How many times in a month do you go out with your s.o.?	How many times in a month do you go out with your s.o.?
How many partners have you had?	How many partners have you had?	How many partners have you had?
How many movies do you see inside a theater in a year?	Number of movies you watched from cinema in the last two weeks	Number of movies you watched from cinema in the last two weeks

Using the the top predictor is sufficient to classify with 89% to 93% accuracy.

Among the three classifiers used, Logistic Regression and Linear SVC yielded the highest average classification accuracy. The table below summarizes the accuracies, and optimal parameters for the three classification models. The minimum required accuracy is 60% (1.25 X PCC).

Model	**Model Parameter**	**Parameter Value**	**Maximum Ave. Accuracy**
kNN	k	3	0.8963
Logistic Regression	C	0.0205	0.9296
Linear SVC	C	0.0068	0.9111

For a more interesting prediction, we reduced the number of features for each model, and also removed the first two features (that give away a person's relationship status):

1. `How many times in a month do you go out with your s.o`
2. `How many partners have you had?`

Using fewer features (though different for each model), we were able to classify with 85% accuracy for kNN, 90% accuracy for logistic regression, and 94% accuracy for SVC.

The results using fewer features are shown below. Linear SVC gives the highest performance with 94% accuracy.

Model	**Model Parameter**	**Parameter Value**	**Maximum Ave. Accuracy**	**No. of Features**
kNN	k	1	0.8481	11
Logistic Regression	C	1.632720	0.9000	9
Linear SVC	C	0.5102	0.9407	13

Below are the features among the reduced that are present in all three models.

Features
Number of movies you watched from cinema in the last two weeks (or in a year for kNN)
How many siblings are you in the family? (not counting half siblings)
What phone brand do you use? Samsung, Apple, Vivo, Huawei, Sony, etc._samsung
What is your favorite number, from 0 to 9?

Aside from frequency of watching cinemas, another interesting feature is the number of siblings in the family. Those in a relationship have on the average more siblings than those not in a relationship. Might it be possible that the number of siblings somehow affects a person's inclination to be in a relationship, e.g. by developing a family-centric attitude? What about the phone brands and even more strangely, favorite number? These are questions that may be tackled by future research.

Take note, however that these findings may not generalize over other populations since the datapoints in this analysis are only from MSc Data Science students at the Asian Institute of Management, which may not represent other populations.

Acknowledgements

I would like to acknowledge Professor Christopher P. Monterola whose Python codes were the foundation of some of the codes used for this analysis, and John Titus Jungao for the suggestion of drawing the PCC on the accuracy vs parameter plot to make it easier to compare with the results of the models.