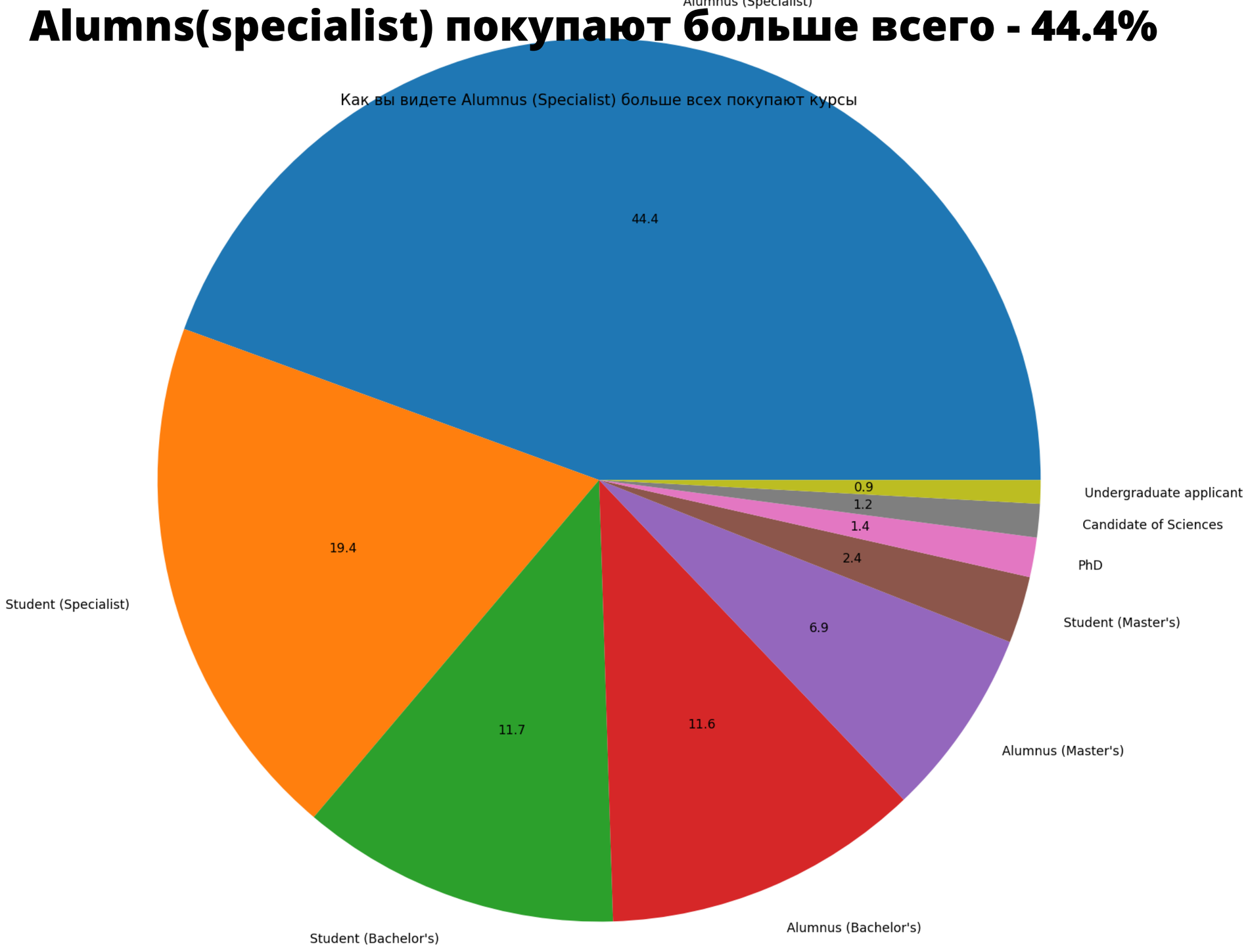
Гипотеза - влияет ли уровень знаний на покупку курсов?

Разработка и проверка гипотезы:

```
fig = plt.figure(dpi = dpi, figsize = (1000 / dpi, 800 / dpi) )#размер и размер текста

temp.plot(kind = 'pie', label = '', title = 'Как вы видете Alumnus (Specialist) больше всех покупают курсы',autopct='%.1f',radius = 1.5)#autopct='%.1f' = задаёт проценты
plt.show()

#clear

df.drop(['bdate','id','has_photo','city','followers_count','occupation_name','last_seen','relation','people_main','life_main','graduation','career_end','career_start','has_mobile'],axis = 1, inplace = True)

#sex

def sex_apply(sex):
    if sex ==1:
        return 0
    return 1
df['sex'] = df['sex'].apply(sex_apply)

#education form
df['education_form'].fillna('Full-time', inplace = True)
df[list(pd.get_dummies(df['education_form']).columns)] = pd.get_dummies(df['education_form'])
df.drop(['education_form'], axis = 1, inplace = True)

#education status
def edu_status_apply(edu_status):
    if edu_status == 'Undergraduate applicant':
        return 0
    elif edu_status == 'Student (Specialist)' or edu_status == "Student (Bachelor's)" or edu_status == "Student (Master's)":
        return 1
    elif edu_status == "Alumnus (Bachelor's)" or edu_status == "Alumnus (Master's)" or edu_status == 'Alumnus (Specialist)':
        return 2
    else:
        return 3
df['education_status'] = df['education_status'].apply(edu_status_apply)

#langs

def Langs_apply(langs):
    if langs.find('Русский') != -1 and langs.find('English') != -1:
        return 0
    else:
        return 1
```

Alumns(specialist) покупают больше всего - 44.4%



Математическая модель

```
df.drop(['education_form'], axis = 1, inplace = True)

#education status
def edu_status_apply(edu_status):
    if edu_status == 'Undergraduate applicant':
        return 0
    elif edu_status == 'Student (Specialist)' or edu_status == "Student (Bachelor's)" or edu_status == "Student (Master's)":
        return 1
    elif edu_status == "Alumnus (Bachelor's)" or edu_status == "Alumnus (Master's)" or edu_status == 'Alumnus (Specialist)':
        return 2
    else:
        return 3
df['education_status'] = df['education_status'].apply(edu_status_apply)

#langs

def Langs_apply(langs):
    if langs.find('Русский') != -1 and langs.find('English') != -1:
        return 0
    else:
        return 1
df['langs'] = df['langs'].apply(Langs_apply)

#occupation type

df['occupation_type'].fillna('university',inplace = True)
def occupation_type_apply(ocu_type):
    if ocu_type == 'university':
        return 0
    return 1
df['occupation_type'] = df['occupation_type'].apply(occupation_type_apply)

#модель

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score

X = df.drop('   result',axis = 1)
y = df['   result']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.40)

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

classifier = KNeighborsClassifier(n_neighbors = 5)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
print(y_test)
print(y_pred)
print('Процент правильно предсказанных исходов:', round(accuracy_score(y_test, y_pred) * 100, 2))
```

Математические модели — это модели, построенные с помощью формул и математических понятий.

Очистка данных

Математическая модель работает только с числовыми данными.

'Выбрасывает Siries'

```
df.drop(['bdate','id','has_photo','city','followers_count','occupation_name','last_seen','relation','people_main','life_main','graduation','career_end','career_start','has_mobile'],axis = 1, inplace = True)
```

Производим очистку оставшиеся данных

```
#sex

def sex_apply(sex):
    if sex ==1:
        return 0
    return 1
df['sex'] = df['sex'].apply(sex_apply)

#langs

def Langs_apply(langs):
    if langs.find('Русский') != -1 and langs.find('English') != -1:
        return 0
    else:
        return 1
df['langs'] = df['langs'].apply(Langs_apply)
```

```
#education form
df['education_form'].fillna('Full-time', inplace = True)
df[list(pd.get_dummies(df['education_form']).columns)] = pd.get_dummies(df['education_form'])
df.drop(['education_form'], axis = 1, inplace = True)
#education status
def edu_status_apply(edu_status):
    if edu_status == 'Undergraduate applicant':
        return 0
    elif edu_status == 'Student (Specialist)' or edu_status == "Student (Bachelor's)" or edu_status == "Student (Master's)":
        return 1
    elif edu_status == "Alumnus (Bachelor's)" or edu_status == "Alumnus (Master's)" or edu_status == 'Alumnus (Specialist)':
        return 2
    else:
        return 3
df['education_status'] = df['education_status'].apply(edu_status_apply)
#occupation type

df['occupation_type'].fillna('university',inplace = True)
def occupation_type_apply(ocu_type):
    if ocu_type == 'university':
        return 0
    return 1
df['occupation_type'] = df['occupation_type'].apply(occupation_type_apply)
```

Составление Математической модели

Импортируем библиотеки

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
```

Составляем тренировочный набор и тестовый

```
X = df.drop('   result',axis = 1)
y = df['   result']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.40)
```

Берём только 40% процентов чтобы модель не запуталась

Составляем модель

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Используем метод 'соседей'

```
classifier = KNeighborsClassifier(n_neighbors = 5)
classifier.fit(X_train, y_train)
```

Тем самым модель может выбрать наиболее подходящий результат

ВЫВОД:

```
y_pred = classifier.predict(X_test)
print(y_test)
print(y_pred)
print('Процент правильно предсказанных исходов:', round(accuracy_score(y_test, y_pred) * 100, 2))
```

```
3844    0
3680    1
5424    0
8192    1
3382    0
       ..
5599    1
4791    1
1552    1
2992    0
4831    0
Name:    result, Length: 3278, dtype: int64
[0 1 0 ... 0 0 0]
Процент правильно предсказанных исходов: 81.79
```

Конец

Спасибо за просмотр и за участие