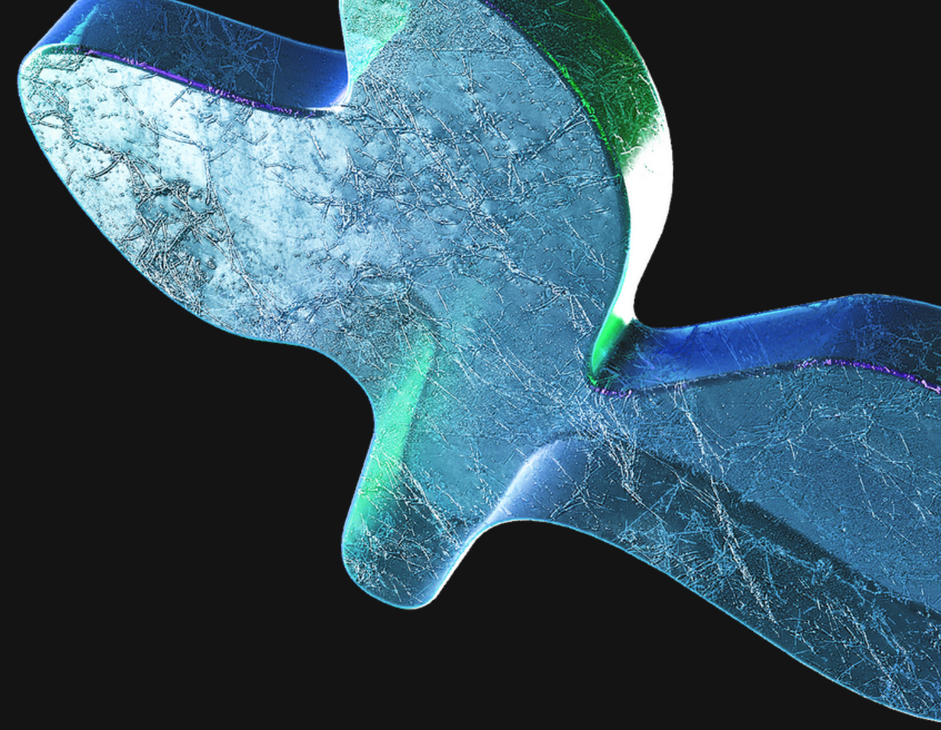
Обучение модели
Digital_edu

Digital_edu

Проверка гипотезы



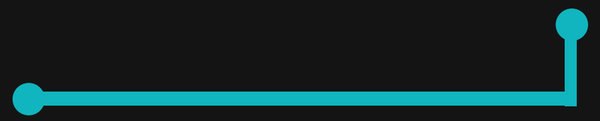
Начало работы

Импорт необходимых модулей / Загрузка данных

Импорт библиотеки sklearn, предназначенная для машинного обучения

```
#Гипотеза 4: Люди, живущие в крупных городах, с большей вероятностью купят курс.
import pandas as pd
import matplotlib.pyplot as plt
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
from sklearn.linear_model import LogisticRegression
df = pd.read_csv('train.csv')
```

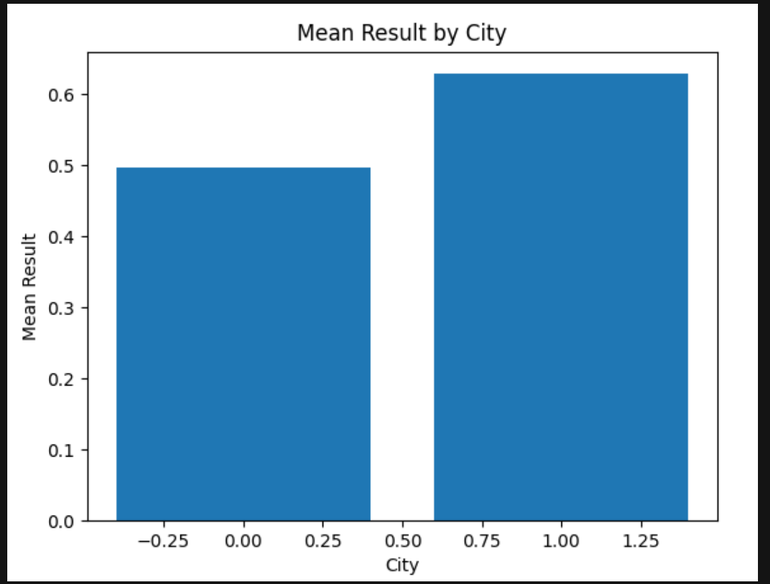
Имортируем и загружаем данные с помощью библиотеки pandas.



Следующий шаг

Вычисление городов и получение статистики

```python
def city_cleaner(city):
    if city in ['Moscow', 'Saint Petersburg', 'Kazan', 'Nur-sultan', 'Yekaterinburg']:
        return 1
    return 0

df['city'] = df['city'].apply(city_cleaner)
city_means = df.groupby(by='city')['result'].mean()

plt.bar(city_means.index, city_means.values)
plt.xlabel('City')
plt.ylabel('Mean Result')
plt.title('Mean Result by City')
plt.show()
```



При выводе получаем среднее значение для платного и бесплатного обеда и предмета.

Группировка и разделение данных по типу обучение и учереждений

Этот код преобразует некоторые столбцы входных данных (которые, представляют собой таблицу данных) с помощью определенных функций.

```
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)


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

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)

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)

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)

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)
```

Функция sex_apply заменяет значения столбца "sex" на 0 для значений, равных 1, и на 1 для всех остальных значений.

Столбец "education_form" заполняется отсутствующими значениями "Full-time". Затем создается несколько новых столбцов, используя функцию pd.get_dummies(), которая создает отдельный столбец для каждого уникального значения в столбце "education_form". Эти столбцы затем добавляются в исходный DataFrame, а столбец "education_form" удаляется.

Функция edu_status_apply преобразует значения столбца "education_status" в числовые значения в соответствии с заданными условиями.

Функция Langs_apply заменяет значения столбца "langs" на 0 для тех строк, где оба языка (русский и английский) указаны в столбце "langs", и на 1 для всех остальных строк.

Столбец "occupation_type" заполняется отсутствующими значениями "university". Затем функция occupation_type_apply заменяет значения столбца "occupation_type" на 0 для значений, равных "university", и на 1 для всех остальных значений.

Все эти преобразования могут быть полезными для анализа данных, например, для обработки отсутствующих значений, категоризации категориальных переменных или преобразования текстовых данных в числовые значения.

Обучение модели при помощи sklearn [kNN]

Построение графиков для средних значений оценок каждого пола и предмета

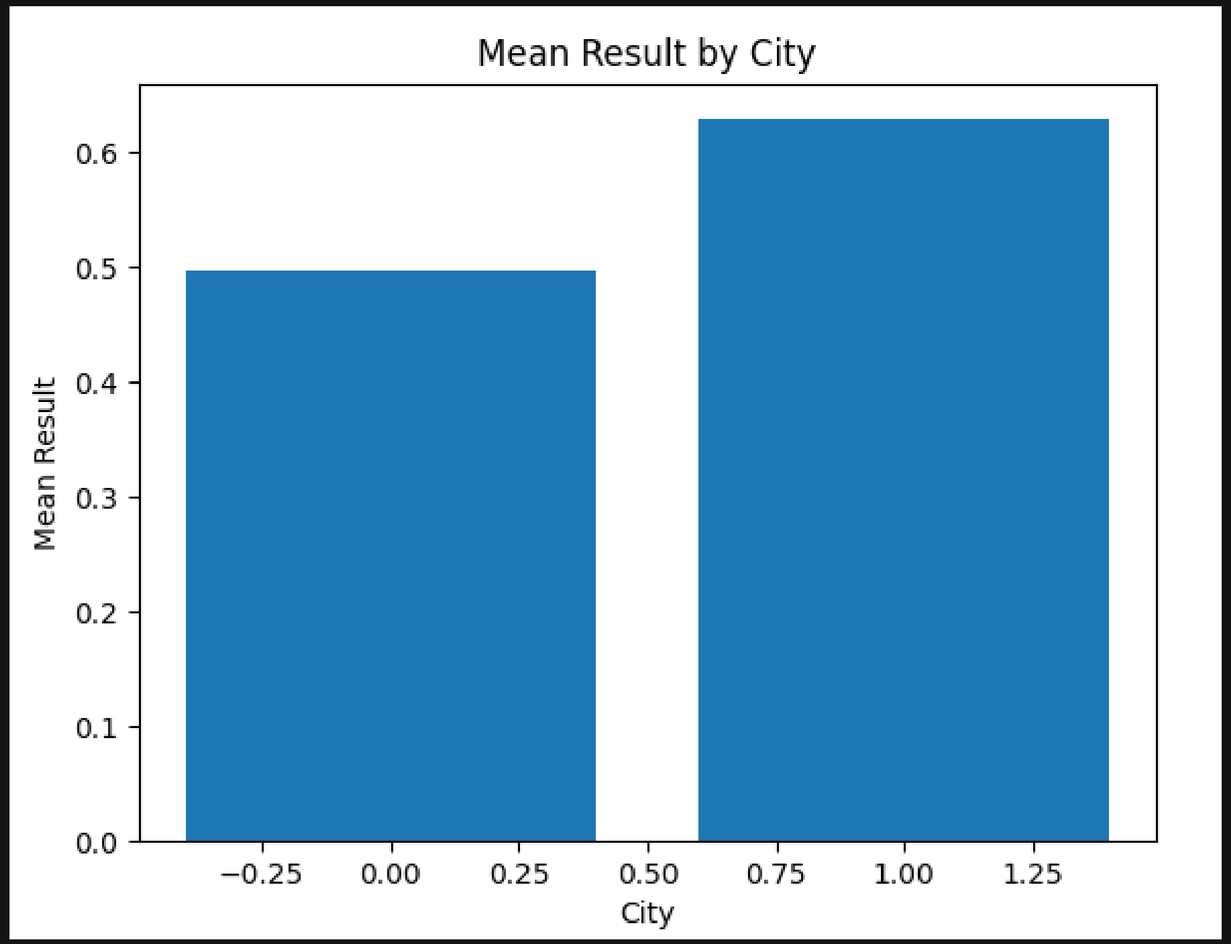
```
# Разделение набора данных на обучающий и тестовые наборы
X = df.drop(['result'], axis=1)
y = df['result']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Масштабироывть объекты с помощью StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Обученик модели KNN
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)

# Прогнозы на тестовом наборе
y_pred = knn.predict(X_test_scaled)

print(confusion_matrix(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
```

Матрица неточностей (англ. Confusion Matrix) — это таблица или диаграмма, показывающая точность прогнозирования классификатора в отношении двух и более классов. Прогнозы классификатора находятся на оси X, а результат (точность) — на оси Y.

Метод ближайших соседей (kNN - k Nearest Neighbours) - метод решения задач классификации и задач регрессии, основанный на поиске ближайших объектов с известными значения целевой переменной.

Результат сей чудо

Вывод данных



Точность прогнозирования классификатора

```
[[655 132]
 [194 658]]
0.8010982306284 32
```

Точность