Project Report

Project Title	Predictive Analysis Using Machine Learning Models
Module Name	WSQ Principles of Machine Learning (SF)
Qualification Name	Advanced Certificate in Data Science (E-Learning)

Student name	**Assessor name**
MEGARAJ MRITTIKA	Ryan Suryanto

Date issued	**Completion date**	**Submitted on**
21.12.2023	11.01.2024	11.01.2024

Project Title	Predictive Analysis using Machine Learning Models

Learner declaration

I certify that the work submitted for this assignment is my own and that research sources are fully acknowledged.

Student signature: Megaraj Mrittika Date: 11.01.2024

Content

Sr. No.	Description	Page No.
1	Project Overview	
2	Project Technical Environment	
3	Analytical Technique & Tools used	
4	Data Science Project Team – Roles and Responsibilities Table	
5	Activity 1: Activity Summary	
6	Activity 2: An Overview	
7	Screenshots of each task of Activity 3	
8	Screenshots of each task of Activity 4	
9	Screenshots of each task of Activity 5	
10	Screenshots of each task of Activity 6	
11	Screenshots of each task of Activity 7	
12	Conclusions and Future Improvements	
10	Appendices A. Code Snippets and Configurations B. Additional Visualizations / Screenshots	

1. Project Overview

Our goal in this modular project is to investigate the extensive and varied OKCupid dataset, which is a goldmine of data on people's interests, lifestyle selections, and personal characteristics. The main goals are to perform a thorough investigation and use predictive modeling for three different categories: income prediction, physical type, and zodiac sign. We also want to investigate a sex prediction task that takes money and education into account.

Project Definition:

Our project starts with a meticulous data loading and exploration phase, aiming to comprehend the dataset's structure, visualize key variables such as age, height, and income distribution, and explore categorical variables like gender, body type, diet, drinks, drugs, and education. Following this, data cleaning and preparation become pivotal, involving handling missing values, converting categorical variables into numeric form, and preparing the dataset for subsequent analyses.

Project Goal:

The main objective is to use machine learning models to forecast outcomes. To forecast the zodiac sign, we will pick pertinent features, pick suitable models (ANN, Decision Tree, SVC, Random Forest), train the models using historical data, assess their effectiveness, and come to some perceptive conclusions. Tasks predicting income and body type will use similar approaches.

An extra degree of complexity and significance is added to the research by including a unique sex prediction challenge based on wealth and educational attainment. Our objective is to obtain precise forecasts and understanding of the correlation between gender, income, and education on the OKCupid platform using a sequence of tasks that comprise data segmentation, training, and assessment.

At the project's conclusion, a comprehensive model comparison and evaluation phase is conducted, during which the accuracy, precision, recall, and F1-score of several models are carefully examined. The complete coding in a Jupyter Notebook, a thorough project report that highlights important discoveries, insights, and model performance, and presentation slides that are intended to facilitate productive stakeholder engagement are among the final deliverables.

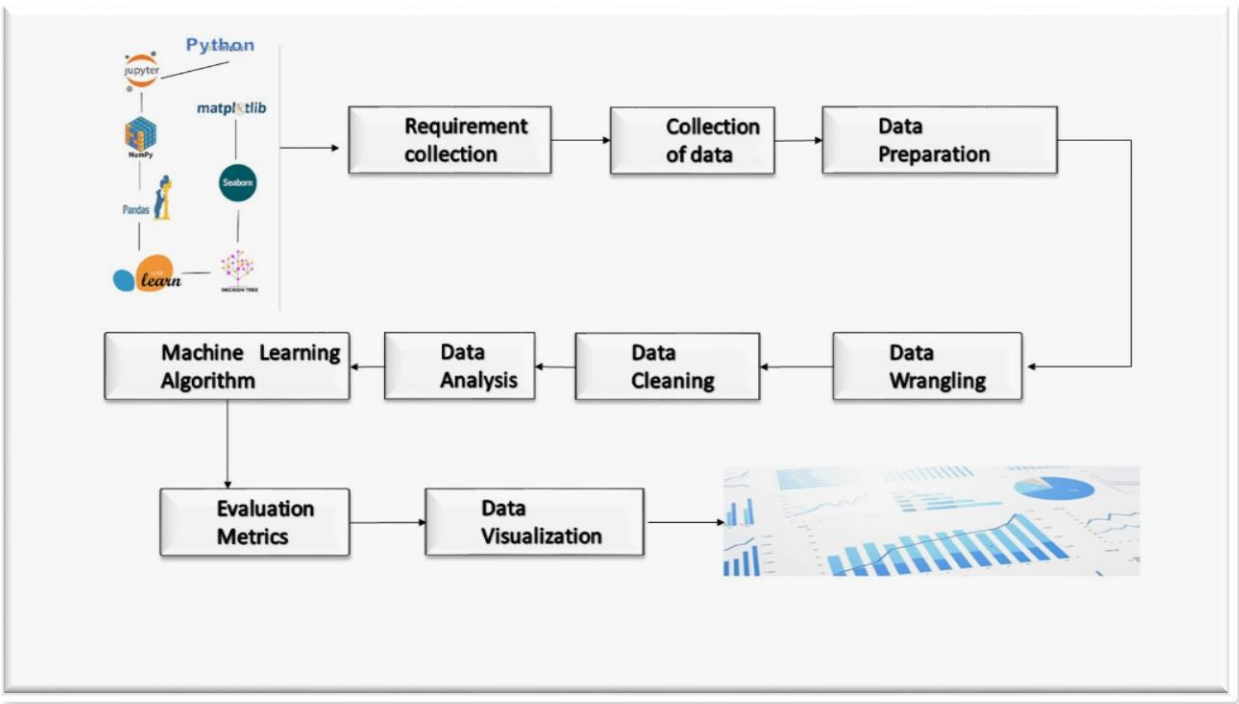
Our ultimate goal is to contribute significant insights into the variables impacting Zodiac sign, body type, wealth, and gender forecasts on OKCupid in addition to offering correct predictions. We want to improve the general understanding of user preferences and behaviors in online dating scenarios by providing practical advice through this thorough analysis, for both individuals and the platform.

2. Project Technical Environment

In the OkCupid project, a highly developed machine learning architecture is at the forefront. This architecture has been painstakingly created to extract significant insights from the enormous and varied OkCupid dataset. This section clarifies the subtleties of the machine learning framework, highlighting its essential elements and the resources that enable it to function.

Architecture for Predictive Analysis in Machine Learning:

The OkCupid project makes use of a powerful machine learning architecture to interpret and forecast several characteristics, such as physical types, income levels, sex depending on education and income, and zodiac signs. The meticulous construction of this machine-learning framework guarantees a methodical and efficient approach to model generation and analysis.



Requirements Collection:

Before diving into the technical nuances of machine learning, a clear understanding of project goals, stakeholders, and data sources is paramount. This initial step lays the groundwork for a focused and effective approach to developing machine learning models.

Python: Serving as the primary programming language, Python provides an expansive and flexible environment for implementing diverse machine learning algorithms and frameworks.

Jupyter: The collaborative and interactive nature of Jupyter notebooks is harnessed for transparent documentation of the entire machine-learning workflow. This fosters clear communication and knowledge sharing among team members.

Certainly! Let's delve into the details of the key packages and modules that will be utilized in the OkCupid project for machine learning tasks.

Pandas :

Overview*:* Pandas is a powerful data manipulation and analysis library for Python. It provides data structures like DataFrames and Series, making it easy to handle and manipulate structured data.

Role in the Project: Pandas will be used for loading, exploring, and manipulating the OkCupid dataset. It facilitates tasks such as handling missing values, converting categorical variables, and preparing data for machine learning.

NumPy :

Overview: NumPy is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these arrays.

Role in the Project: NumPy will serve as the backbone for efficient numerical operations and handling arrays, essential for the underlying mathematical computations in machine learning models.

Matplotlib:

Overview: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. pyplot is a sub-library that provides a MATLAB-like interface for creating plots and charts.

Role in the Project: Matplotlib. pyplot will be used for visualizing data distributions, relationships, and model performance through various charts and plots.

Seaborn:

Overview: Seaborn is a statistical data visualization library built on top of Matplotlib.

It provides a high-level interface for drawing attractive and informative statistical graphics.

Role in the Project: Seaborn complements Matplotlib, enhancing the aesthetics of visualizations and simplifying the creation of complex statistical plots.

Scikit-Learn:

Overview: Scikit-Learn is a machine-learning library that provides simple and efficient tools for data analysis and modeling. It includes various algorithms for classification, regression, clustering, and more.

Modules Used:

- ***train_test_split***: Used for splitting the dataset into training and testing sets.
- ***GridSearchCV***: Implements grid search cross-validation to tune hyper parameters.
- ***MinMaxScaler***: Scales features to a specified range, commonly used for neural networks.
- **MLPClassifier**: Implements a Multi-Layer Perceptron classifier (neural network).
- ***DecisionTreeClassifier:*** Implements a decision tree classifier.
- ***SVC***: Implements a Support Vector Classifier.
- ***RandomForestClassifier***: Implements a Random Forest classifier.
- ***Classification_report***: Evaluate and report the performance of a classifier.

Role in the Project: Scikit-Learn will be the core tool for implementing machine learning models, tuning hyperparameters, and evaluating model performance using classification metrics.

These packages collectively form a powerful and comprehensive toolkit for the OkCupid project, facilitating tasks ranging from data preprocessing to the training and evaluation of machine learning models.

Data Collection:

To obtain a deep understanding of the OkCupid dataset's structure and the distribution of its important characteristics, the first step in the journey is to load and explore the dataset. This dataset is used to extract real-world preferences, lifestyle decisions, and personal characteristics, providing a rich and varied set of features for research.

Data Preparation:

Managing missing values, transforming categorical variables into numerical form, and taking care of data cleaning and preprocessing chores are all necessary to guarantee that the dataset is ready for machine learning models. This crucial stage creates a consistent, clean dataset that is ready to train machine learning models.

Machine Learning Model Implementation:

The OkCupid project uses a variety of machine learning methods to address particular prediction tasks after preparing the dataset. Every algorithm is different from the others and is chosen according to the type of prediction.

An outline of the main machine learning algorithms in use is provided below:

1. The Multi-Layer Perceptron (MLP) Classifier: For intricate pattern identification, this kind of artificial neural network is used. With its many layers of nodes (neurons), it can record complex

relationships in the data. When it comes to tasks like predicting a person's zodiac sign and sex based on income and education level, MLP is especially useful.

2. Decision Tree Classifier: Recursively dividing the dataset according to feature values, decision trees are simple models. They do well in activities such as body type prediction. Decision trees are useful for comprehending the factors impacting predictions since they are simple to read and visualize.

3. Support Vector Classifier (SVC): SVC works by determining which hyperplane best divides data points from various classes. It works well for assignments like income prediction, where the objective is to categorize examples into various salary brackets. SVC handles non-linear interactions in data effectively.

4. Random Forest Classifier: During training, the Random Forest ensemble learning technique creates a number of decision trees. It does exceptionally well at reducing overfitting and raising prediction accuracy. It uses the power of numerous decision trees to predict Zodiac signs, among other things, in the OkCupid project.

Specifics of Implementation:

- *Feature Selection:* To improve model performance and interpretability, pertinent features are carefully chosen for each prediction assignment.
- *Hyperparameter Tuning:* To ensure ideal model configurations, GridSearchCV is employed to methodically investigate hyperparameter combinations.
- *Training and Evaluation:* The test set is used to assess the models' performance after they have been trained on the training dataset. Metrics for evaluation including accuracy, precision, recall, and F1-score give a thorough picture of the advantages and disadvantages of each model.

Model Comparison and Evaluation:

A thorough comparison of several machine learning models marks the project's conclusion. Evaluation metrics are used to evaluate each model's performance, including accuracy, precision, recall, and F1-score. Each model's overall effectiveness is taken into consideration while identifying its strengths and drawbacks and offering recommendations.

This machine learning architecture guarantees a logical and efficient approach to collecting significant insights from the OkCupid dataset, ultimately helping to the successful implementation of predictive analytic tasks. It is effortlessly incorporated into the technical environment of the project.

3. Analytical Techniques & Tools used

1. Data Loading and Exploration:

Techniques: Comprehensive exploratory data analysis (EDA), statistical analysis, and visualization.

Tools: Leveraged Python libraries including Pandas, NumPy, and Matplotlib for loading, exploring, and visualizing the OKCupid dataset. Jupyter Notebooks provided an interactive environment for step-by-step analysis.

2. Data Cleaning and Preparation:

Techniques: Handling missing values, converting categorical variables into numeric form, and thorough data cleaning and preprocessing.

Tools: Utilized Pandas and custom Python scripts for efficient removal of duplicates, imputation of missing values, and meticulous data cleaning.

3. Zodiac Sign Prediction:

Techniques: Feature selection, machine learning model selection, training, and evaluation.

Tools: Employed machine learning models such as Artificial Neural Networks (ANN), Decision Trees, Support Vector Classifier (SVC), and Random Forest for predicting Zodiac signs. Leveraged Scikit-Learn and evaluated model performance using metrics like accuracy and F1-score.

4. Body Type Prediction:

Techniques: Feature selection, machine learning model selection, training, and evaluation.

Tools: Utilized machine learning models like ANN, Decision Tree, SVC, and Random Forest for predicting body types. Employed Scikit-Learn for model training and evaluation, comparing performance across different algorithms.

5. Income Prediction:

Techniques: Investigating and handling missing or invalid values in the income column, feature selection, hyperparameter tuning, and model evaluation.

Tools: Employed Python libraries such as Pandas and Scikit-Learn for investigating and preprocessing the income column. Selected machine learning models and tuned hyperparameters for accurate income prediction. Evaluated and compared model performance using Scikit-Learn metrics.

6. Sex Prediction based on Education Level and Income:

Techniques: Creating a copy of the dataset, handling missing values, converting education into dummy variables, splitting data, training machine learning models, and feature normalization.

Tools: Used Pandas for dataset manipulation, Scikit-Learn for machine learning tasks, and Python for handling data transformations and splitting. Employed machine learning models like ANN, Decision Tree, SVC, and Random Forest for sex prediction.

7. Model Comparison and Evaluation:

Techniques: Comparing performance metrics, evaluating accuracy, precision, recall, and F1-score, and identifying strengths and weaknesses of each model.

Tools: Leveraged Scikit-Learn for comparing and evaluating machine learning models. Summarized results and provided recommendations based on model performance.

In summary, the OkCupid project utilized a combination of Python programming, machine learning techniques, and data exploration tools to extract valuable insights from the dataset, addressing various predictive analysis tasks related to Zodiac signs, body types, income, and sex prediction. The project followed a systematic approach, employing a variety of analytical techniques and tools tailored to each specific task.

4. Data Science Project Team – Roles and Responsibilities

Roles	Responsibilities
Project Manager	• Define project scope, goals, and deliverables. • Develop project plans and timelines. • Allocate resources and coordinate team members. • Monitor and report on project progress. • Ensure the project aligns with business objectives. • Manage project budget and resources effectively. • Mitigate risks and resolve project issues. • Foster collaboration and communication within the team.
Data Scientist	• Analyze and interpret complex datasets. • Develop and implement predictive models. • Design experiments and analyze results. • Collaborate with domain experts to understand business requirements. • Communicate findings to non-technical stakeholders. • Stay current with the latest advancements in data science and machine learning. • Contribute to research and development initiatives.
Data Engineer	• Develop and optimize data architectures. • Build and maintain the infrastructure for data generation, transformation, and storage.

	• Ensure data quality and integrity. • Collaborate with data scientists to deploy models into production. • Implement and maintain ETL (Extract, Transform, Load) processes. • Explore and recommend new technologies for data processing and storage.
Machine Learning Engineer	• Implement machine learning algorithms and models. • Work closely with data scientists and data engineers to integrate models into systems. • Develop scalable and efficient machine learning pipelines. • Monitor and update models for continuous improvement. • Collaborate with IT and DevOps teams for model deployment and monitoring.
Domain Expert / Subject Matter Expert	• Provide industry or domain-specific knowledge. Define business problems and objectives. Collaborate with data scientists to translate business requirements into analytical solutions. • Validate and interpret model results in the context of the business. • Conduct market research to identify emerging trends and opportunities.

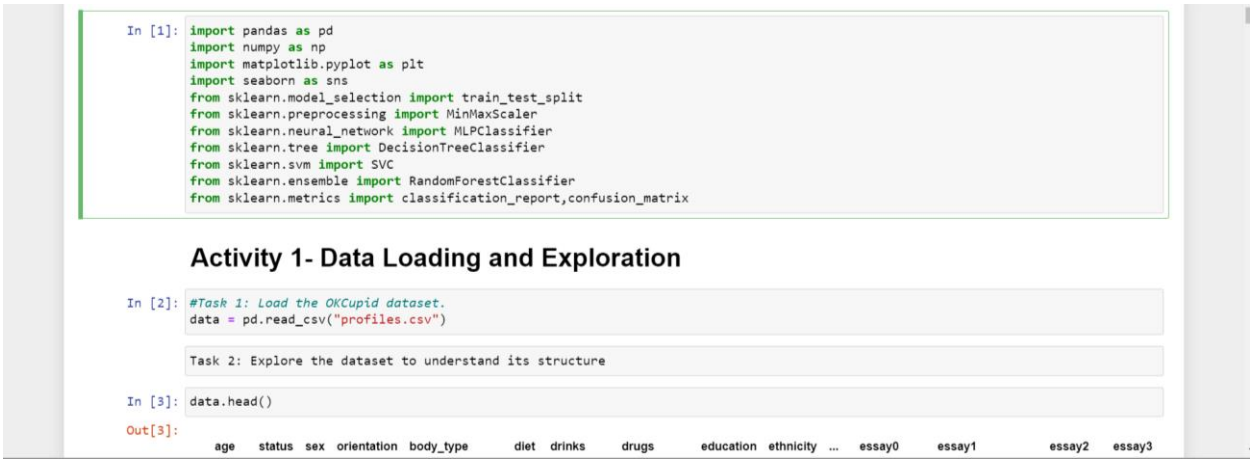
	• Act as a liaison between technical and non-technical teams. • Lead workshops and training sessions to enhance domain knowledge within the team.
Data Analyst	• Collect, clean, and preprocess data for analysis. Perform exploratory data analysis. • Generate reports and visualizations. • Assist in the interpretation of data insights. Collaborate with data scientists to understand data requirements. • Conduct A/B testing and statistical analysis. Automate routine data analysis tasks. • Develop and maintain dashboards for real-time monitoring.
Database Administrator	• Manage and optimize databases. • Ensure data security and integrity. • Troubleshoot database issues. • Design and implement data storage solutions. Collaborate with data engineers to design efficient database schemas. • Implement data backup and recovery strategies. Monitor and optimize database performance.
Software Developer / Engineer	• Develop and maintain software applications that integrate with data science solutions. • Collaborate with data engineers and machine learning engineers for system integration.

	• Ensure the scalability and reliability of software components. • Conduct code reviews and maintain coding standards. • Participate in Agile development processes. Collaborate with UI/UX designers for a seamless user experience.
Business Analyst	• Identify business opportunities and challenges. Translate business needs into data requirements. • Collaborate with project manager and stakeholders to align projects with business goals. • Conduct cost-benefit analysis for proposed solutions. • Evaluate and recommend technology solutions to address business needs. • Identify process improvements and efficiency gains. • Develop and maintain business process documentation.
Data Privacy Officer	• Ensure compliance with data protection regulations (e.g., GDPR). • Develop and implement data privacy policies and procedures. • Educate the team on data privacy best practices.

	• Conduct privacy impact assessments for new projects. • Manage data subject access requests. Collaborate with legal and compliance teams.
IT Security Specialist	• Implement and manage cybersecurity measures for data and systems. • Conduct vulnerability assessments and penetration testing. • Monitor and respond to security incidents. • Ensure compliance with security policies and standards. Collaborate with data and software engineers for secure data handling.
Communication Specialist	• Communicate project updates and findings to stakeholders. • Translate technical information for non-technical audiences. • Facilitate communication between team members with different expertise. • Assist in creating documentation for end-users. • Develop and maintain project communication plans.

5. Activity 1: Data Loading and Exploration

Task 1: Load the OKCupid dataset.



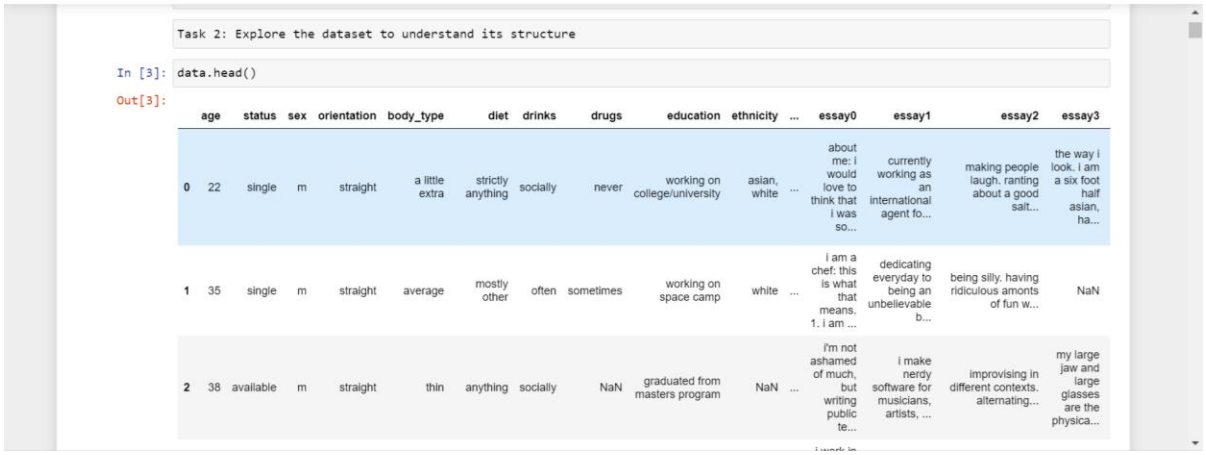
```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import MinMaxScaler
        from sklearn.neural_network import MLPClassifier
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.svm import SVC
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import classification_report,confusion_matrix
```

Activity 1- Data Loading and Exploration

```
In [2]: #Task 1: Load the OKCupid dataset.
        data = pd.read_csv("profiles.csv")

        Task 2: Explore the dataset to understand its structure

In [3]: data.head()
Out[3]:
```

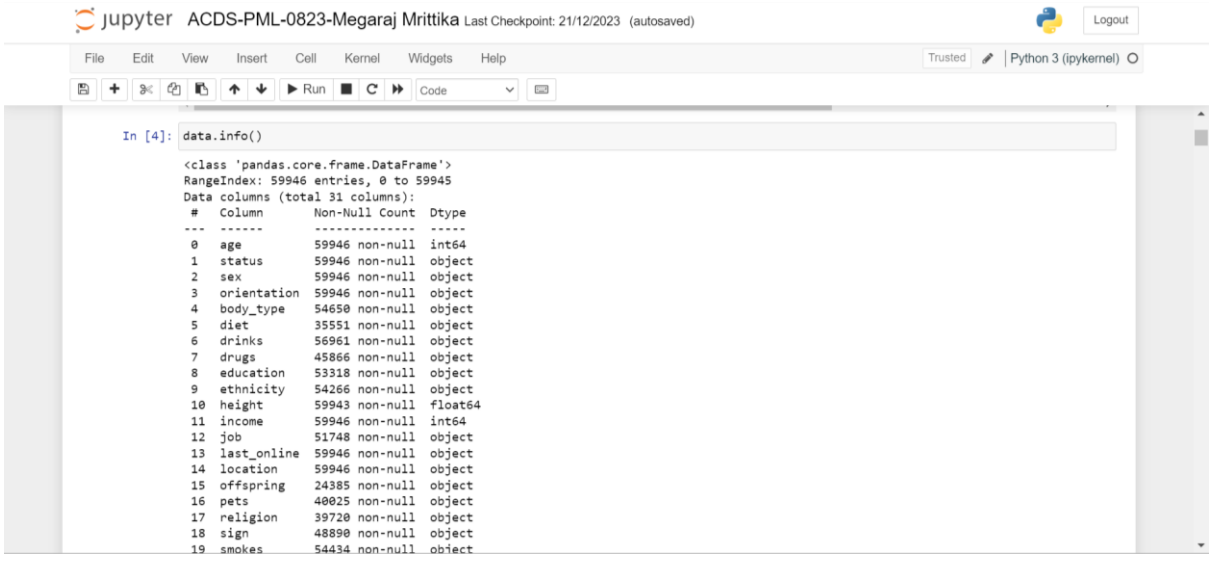
	age	status	sex	orientation	body_type	diet	drinks	drugs	education	ethnicity	...	essay0	essay1	essay2	essay3

Task 2: Explore the dataset to understand its structure.



```
Task 2: Explore the dataset to understand its structure

In [3]: data.head()
Out[3]:
```

	age	status	sex	orientation	body_type	diet	drinks	drugs	education	ethnicity	...	essay0	essay1	essay2	essay3
0	22	single	m	straight	a little extra	strictly anything	socially	never	working on college/university	asian, white	...	about me: i would love to think that i was so...	currently working as an international agent fo...	making people laugh. ranting about a good salt...	the way i look. i am a six foot half asian, ha...
1	35	single	m	straight	average	mostly other	often	sometimes	working on space camp	white	...	i am a chef: this is what that means. 1. i am ...	dedicating everyday to being an unbelievable b...	being silly. having ridiculous amonts of fun w...	NaN
2	38	available	m	straight	thin	anything	socially	NaN	graduated from masters program	NaN	...	i'm not ashamed of much, but writing public te...	i make nerdy software for musicians, artists, ...	improvising in different contexts. alternating...	my large jaw and large glasses are the physica...



jupyter ACDS-PML-0823-Megaraj Mrittika Last Checkpoint: 21/12/2023 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

```
In [4]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59946 entries, 0 to 59945
Data columns (total 31 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   age          59946 non-null  int64
 1   status       59946 non-null  object
 2   sex          59946 non-null  object
 3   orientation  59946 non-null  object
 4   body_type    54650 non-null  object
 5   diet         35551 non-null  object
 6   drinks       56961 non-null  object
 7   drugs        45866 non-null  object
 8   education    53318 non-null  object
 9   ethnicity    54266 non-null  object
 10  height       59943 non-null  float64
 11  income       59946 non-null  int64
 12  job          51748 non-null  object
 13  last_online  59946 non-null  object
 14  location     59946 non-null  object
 15  offspring    24385 non-null  object
 16  pets         40025 non-null  object
 17  religion     39720 non-null  object
 18  sign         48890 non-null  object
 19  smokes       54434 non-null  object
```

```
In [5]: data.shape
Out[5]: (59946, 31)

In [6]: data.describe()
Out[6]:
```

	age	height	income
count	59946.000000	59943.000000	59946.000000
mean	32.340290	68.295281	20033.222534
std	9.452779	3.994803	97346.192104
min	18.000000	1.000000	-1.000000
25%	26.000000	66.000000	-1.000000
50%	30.000000	68.000000	-1.000000
75%	37.000000	71.000000	-1.000000
max	110.000000	95.000000	1000000.000000

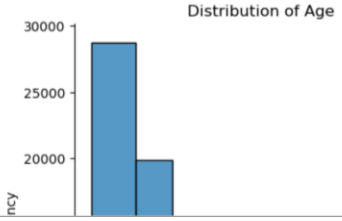
```
In [7]: data.dtypes
Out[7]: age              int64
        status          object
        sex             object
        orientation     object
        body_type       object
        diet            object
        drinks          object
        drugs           object
        education       object
        ethnicity       object
        height         float64
        income           int64
        job             object
        last_online     object
        location        object
        offspring       object
        pets            object
        religion        object
        sign            object
        smokes          object
        speaks          object
        essay0          object
        essay1          object
        essay2          object
        essay3          object

In [8]: data.index
Out[8]: RangeIndex(start=0, stop=59946, step=1)

In [9]: data.columns
Out[9]: Index(['age', 'status', 'sex', 'orientation', 'body_type', 'diet', 'drinks',
               'drugs', 'education', 'ethnicity', 'height', 'income', 'job',
               'last_online', 'location', 'offspring', 'pets', 'religion', 'sign',
               'smokes', 'speaks', 'essay0', 'essay1', 'essay2', 'essay3', 'essay4',
               'essay5', 'essay6', 'essay7', 'essay8', 'essay9'],
              dtype='object')
```

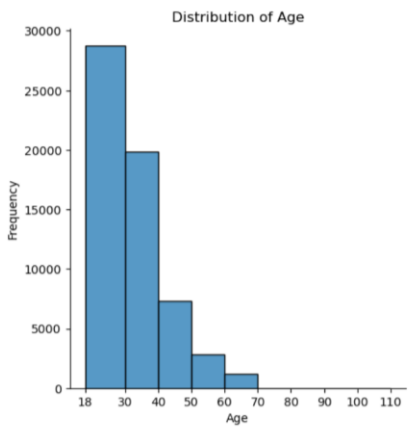
Task 3: Visualize and describe the distribution of key variables like age, height, and income.

```
Task 3: Visualize and describe the distribution of key variables like age, height, and income
```

Age Distribution

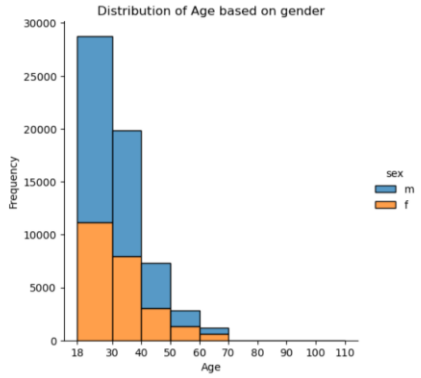
```
In [10]: sns.displot(data=data,x="age",bins=[18,30,40,50,60,70,80,90,100,110])
         plt.xticks([18,30,40,50,60,70,80,90,100,110])
         plt.title('Distribution of Age')
         plt.xlabel('Age')
         plt.ylabel('Frequency')
         plt.show()
```





Age Distribution based on gender

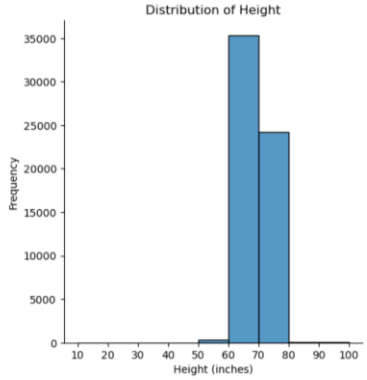
In [11]:
```
sns.displot(data=data,x="age",bins=[18,30,40,50,60,70,80,90,100,110],hue="sex",multiple="stack")
plt.xticks([18,30,40,50,60,70,80,90,100,110])
plt.title('Distribution of Age based on gender')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



According to the histogram, the distribution for both genders are the same, but there are more male users than female users between 18 to 40

Height Distribution

In [12]:
```
sns.displot(data=data,x="height",bins=[10,20,30,40,50,60,70,80,90,100])
plt.xticks([10,20,30,40,50,60,70,80,90,100])
plt.title('Distribution of Height')
plt.xlabel('Height (inches)')
plt.ylabel('Frequency')
plt.show()
```

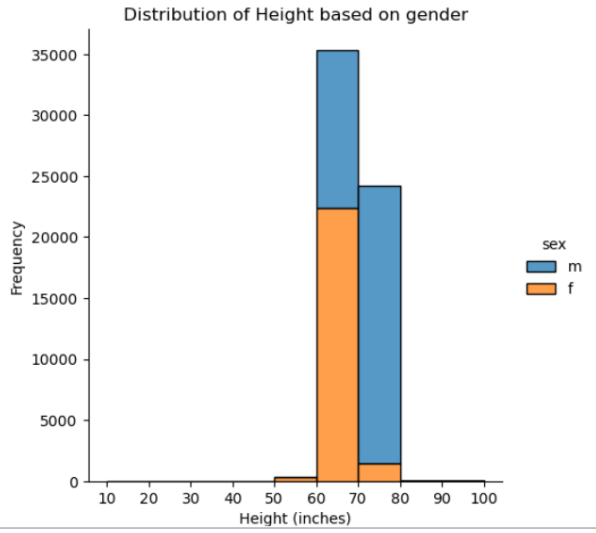


The heights of users has a normal distribution.Heights of most of the users are between 60 and 80

Height Distribution based on gender

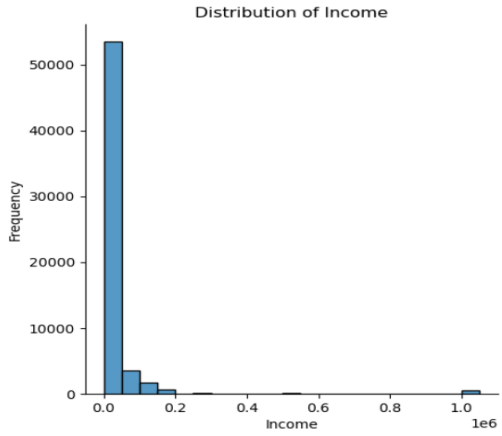
Height Distribution based on gender

In [13]:
```
sns.displot(data=data,x="height",bins=[10,20,30,40,50,60,70,80,90,100],hue="sex",multiple="stack")
plt.xticks([10,20,30,40,50,60,70,80,90,100])
plt.title('Distribution of Height based on gender')
plt.xlabel('Height (inches)')
plt.ylabel('Frequency')
plt.show()
```



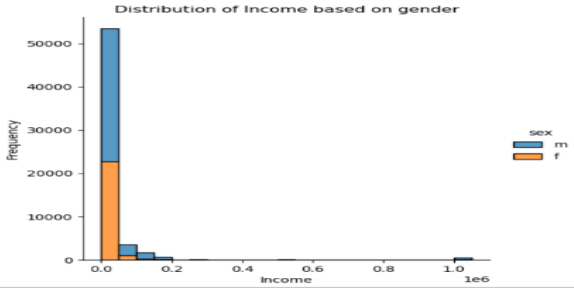
Income Distribution

In [14]:
```
sns.displot(data=data,x="income",binwidth = 50000)
plt.title('Distribution of Income')
plt.xlabel('Income')
plt.ylabel('Frequency')
plt.show()
```



Income Distribution based on gender

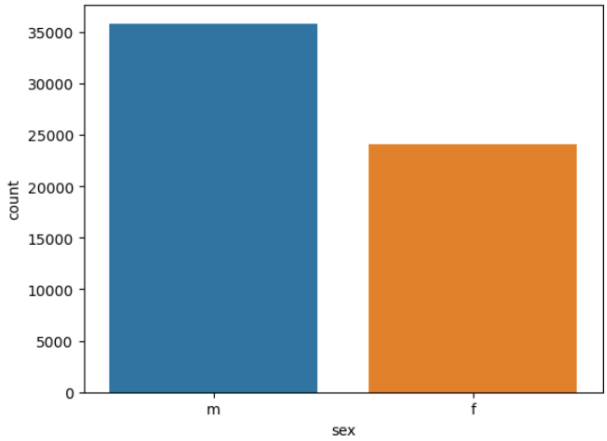
In [15]:
```
sns.displot(data=data,x="income",binwidth = 50000,hue="sex",multiple="stack")
plt.title('Distribution of Income based on gender')
plt.xlabel('Income')
plt.ylabel('Frequency')
plt.show()
```



Task 4: Explore categorical variables such as gender, body type, diet, drinks, drugs, education, etc

Gender

```
In [16]: data.sex.value_counts()
Out[16]: m    35829
         f    24117
         Name: sex, dtype: int64

In [17]: sns.countplot(data=data,x="sex")
         plt.show()
```

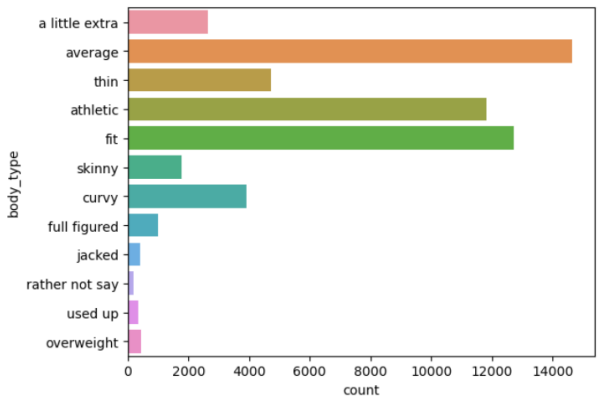


sex

Male users are more than female users.

Body Type

```
In [18]: data.body_type.value_counts()
Out[18]: average           14652
         fit               12711
         athletic          11819
         thin               4711
         curvy              3924
         a little extra     2629
         skinny             1777
         full figured       1009
         overweight          444
         jacked              421
         used up             355
         rather not say      198
         Name: body_type, dtype: int64

In [19]: sns.countplot(data=data,y="body_type")
         plt.show()
```



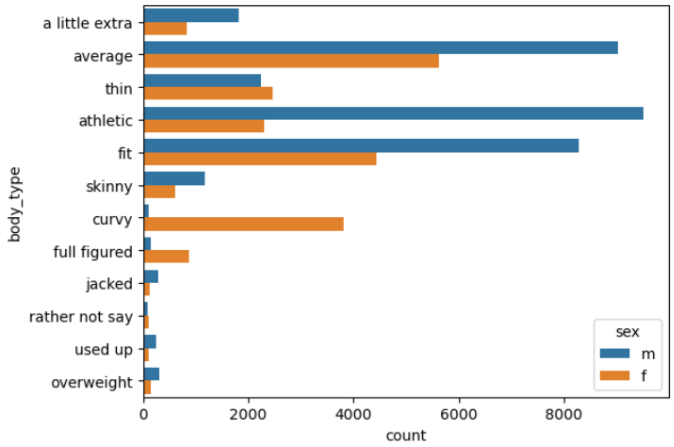
Most of the users declared their body type as average, fit, or athletic.

Body Type based on gender

```
In [20]: sns.countplot(data=data,y="body_type",hue="sex")
         plt.show()
```

Body Type based on gender

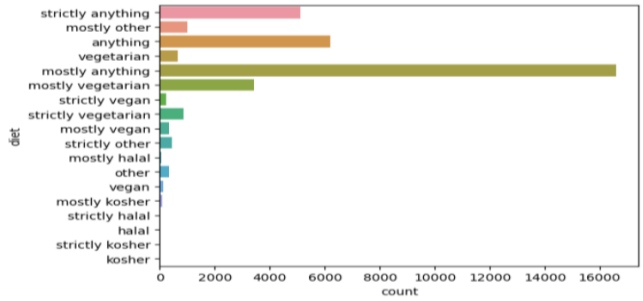
In [20]:
```
sns.countplot(data=data,y="body_type",hue="sex")
plt.show()
```



When we split countplot of body type variable by gender, we can see that most of the females described their body types with following three categories: average, fit or curvy but males described their body types with athletic, average or fit.It seems that curvy and full figured are feminine descriptions and overweight and jacked used mostly by male users.

Diet

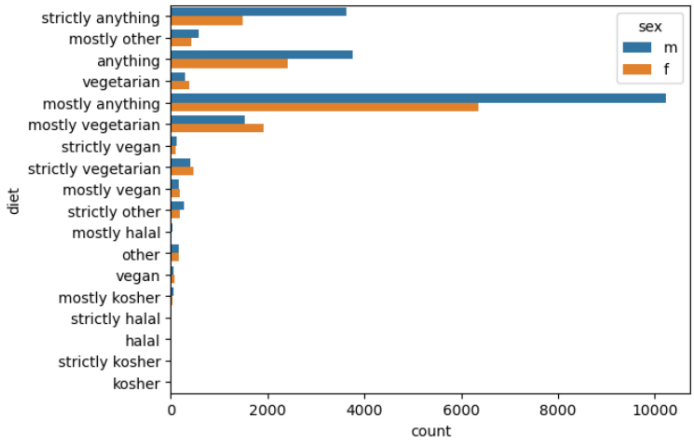
In [21]:
```
sns.countplot(data=data,y="diet")
plt.show()
```



According to the countplot above, most of the users eat mostly anything, followed by 'anything', 'strongly anything'.The fourth popular diet is 'mostly vegeterian'.

Diet based on gender

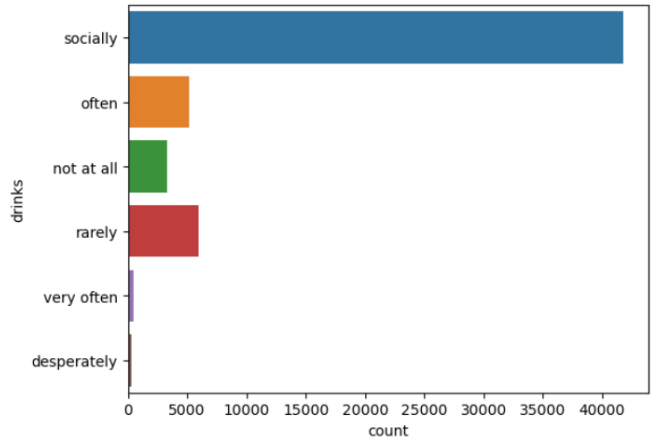
In [22]:
```
sns.countplot(data=data,y="diet",hue="sex")
plt.show()
```



When we split the diet data with gender, women are more likely to be vegetarian.

Drinks

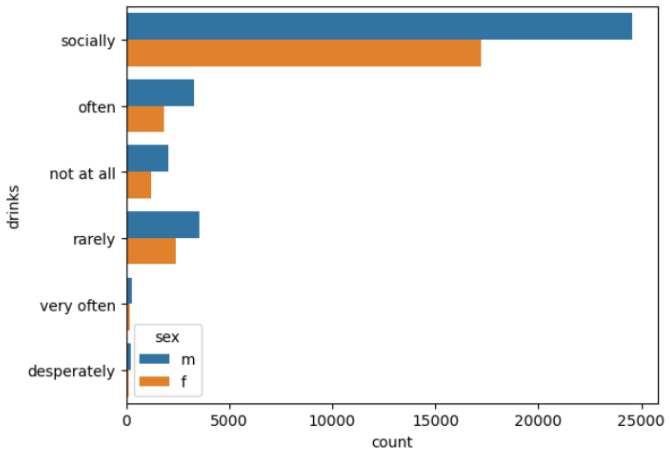
In [23]:
```
sns.countplot(data=data,y="drinks")
plt.show()
```



The majority of the users drink "socially".

Drinks based on gender

In [24]:
```
sns.countplot(data=data,y="drinks",hue="sex")
plt.show()
```

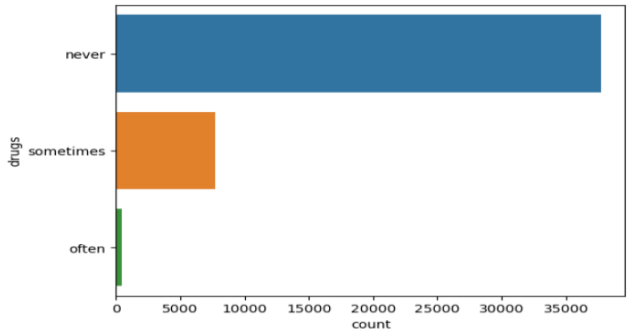


The majority of the users drink "socially" by males than females

---

Drugs

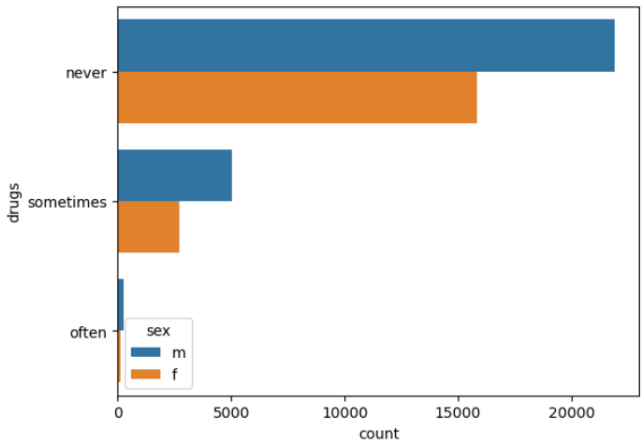
In [25]:
```
sns.countplot(data=data,y="drugs")
plt.show()
```



we can see that overwhelming majority of the users never use drugs.

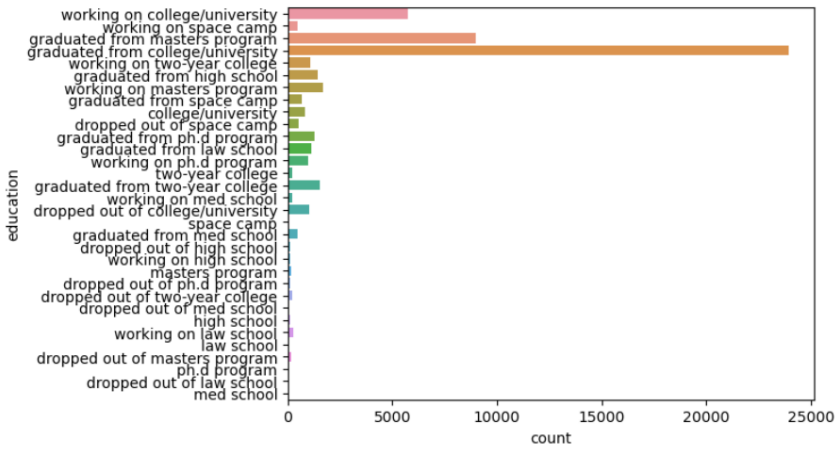
Drugs based on gender

In [26]:
```
sns.countplot(data=data,y="drugs",hue="sex")
plt.show()
```



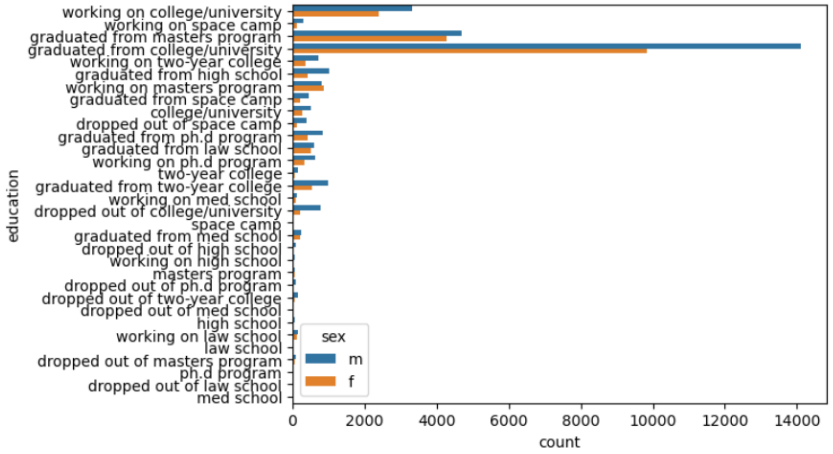
Education

In [27]:
```
sns.countplot(data=data,y="education")
plt.show()
```



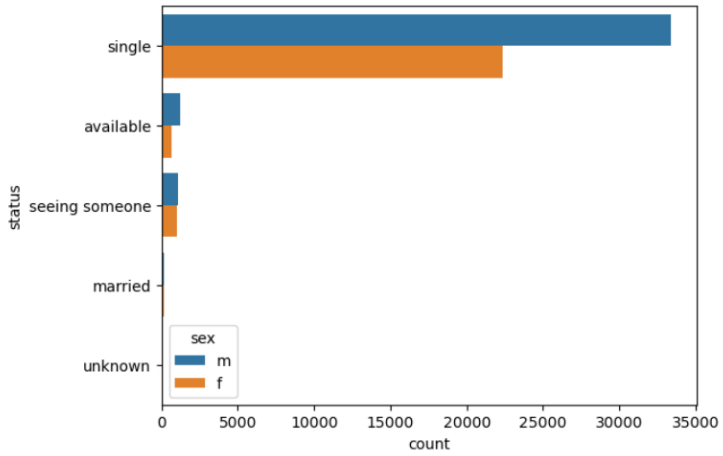
According to the chart Above, we can see that the majority of users graduated from college/university followed by masters graduates and people who 'working on college/university'.

In [28]:
```
sns.countplot(data=data,y="education",hue="sex")
plt.show()
```



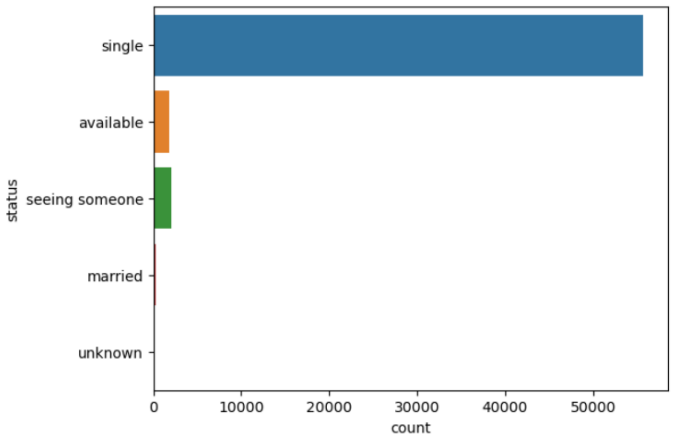
Status based on gender

In [30]:
```
sns.countplot(data=data,y="status",hue="sex")
plt.show()
```



Status

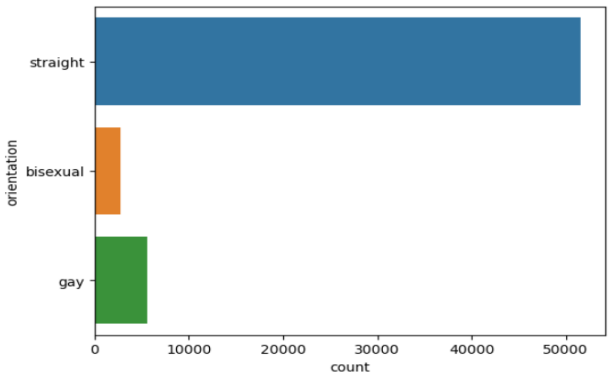
In [29]:
```
sns.countplot(data=data,y="status")
plt.show()
```



The vast majority of uses are single.This is reasonable.Because they registered in this dating app to find a partner.

Orientation

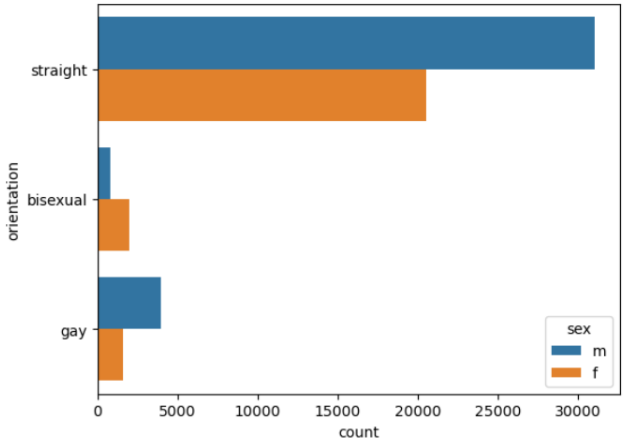
In [31]:
```
sns.countplot(data=data,y="orientation")
plt.show()
```



According to the chart Above, as far as sexual orientation is concerned, the majority of the users are straight.

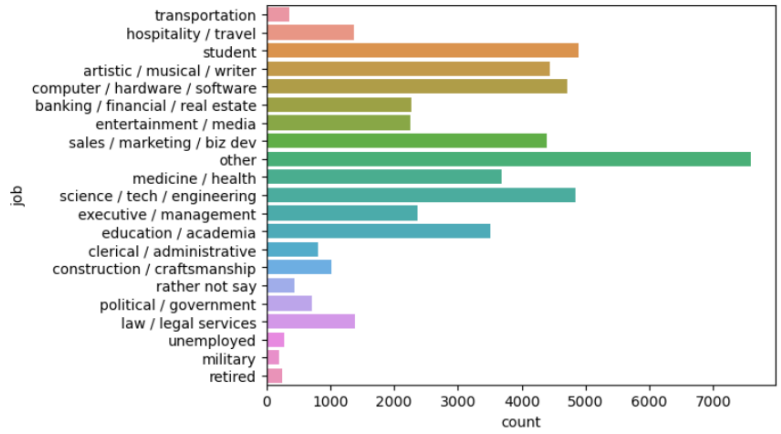
Orientation based on gender

In [32]:
```
sns.countplot(data=data,y="orientation",hue="sex")
plt.show()
```



Job

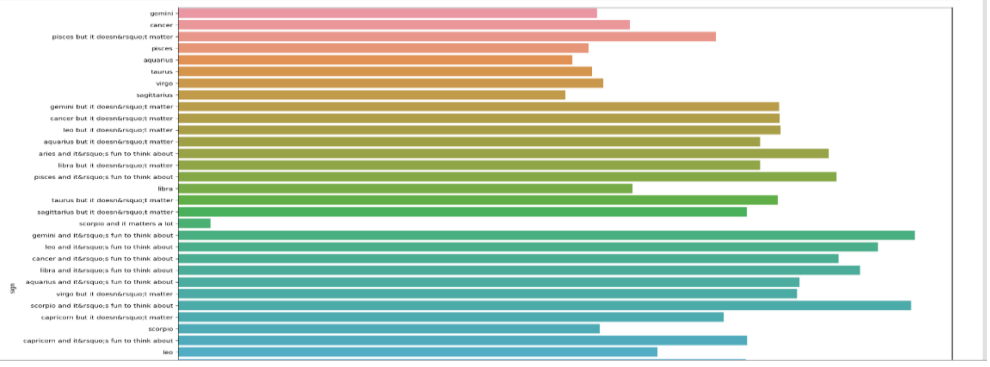
In [33]:
```
sns.countplot(data=data,y="job")
plt.show()
```



Students, technicians, computer specialists, artists and those who works in business make up the majority of OKCupid users with roughly the equal percentage of users.

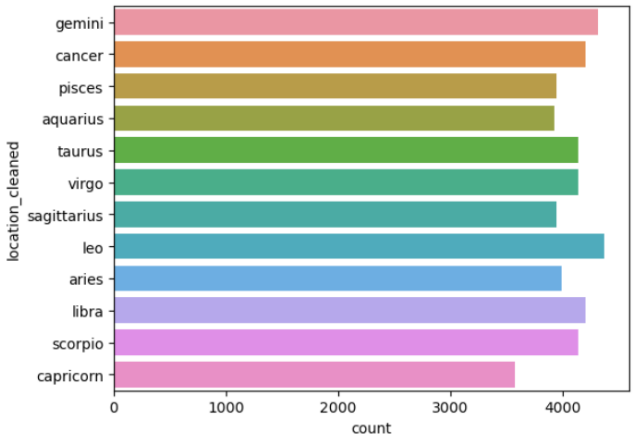
location

In [34]:
```
plt.figure(figsize=(20,20))
sns.countplot(data=data,y="sign")
plt.show()
```



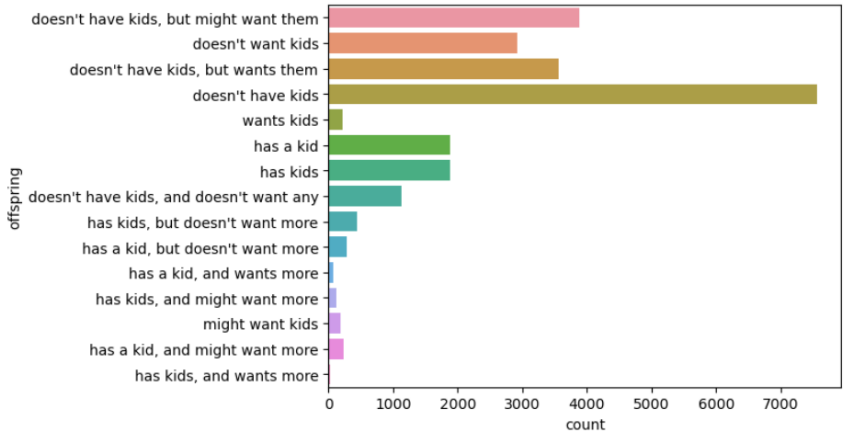
In [35]:
```
data['location_cleaned']=data.sign.str.split().str[0]
```

In [36]:
```
sns.countplot(data=data,y="location_cleaned")
plt.show()
```



Offspring

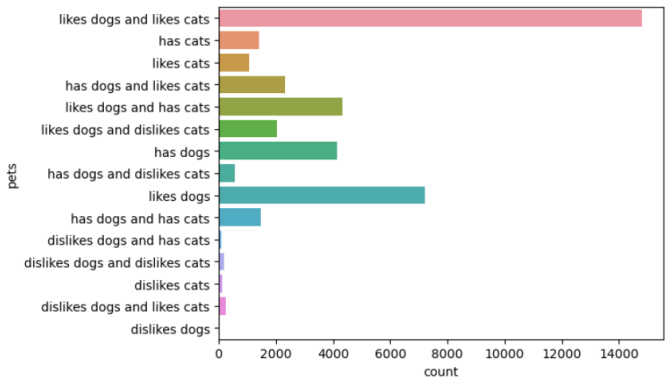
In [37]:
```
sns.countplot(data=data, y="offspring")
plt.show()
```



The data suggest that most users do not have kids.

Pets

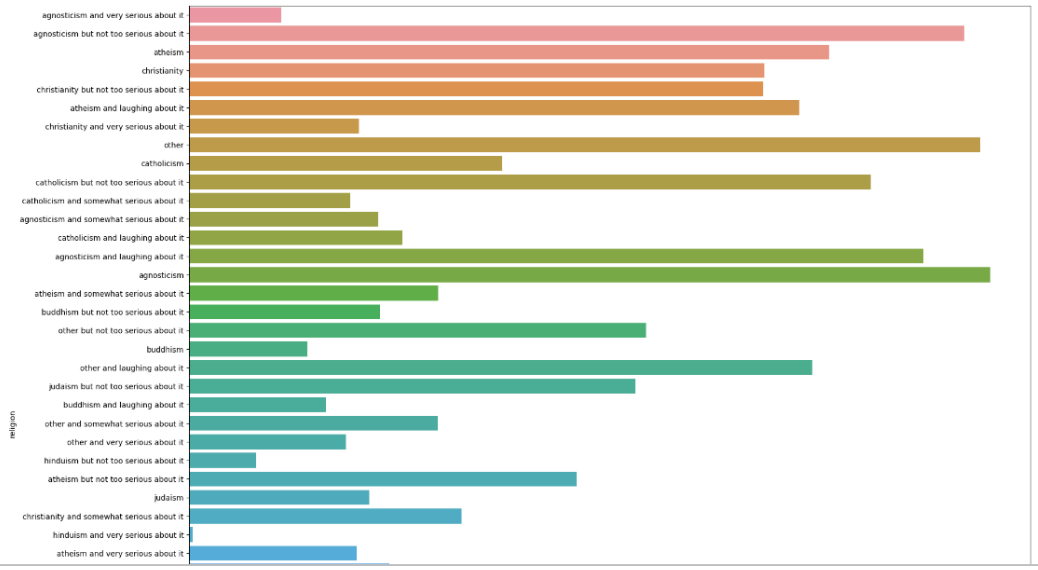
In [38]:
```
sns.countplot(data=data, y="pets")
plt.show()
```



Most users like both dogs and cats.

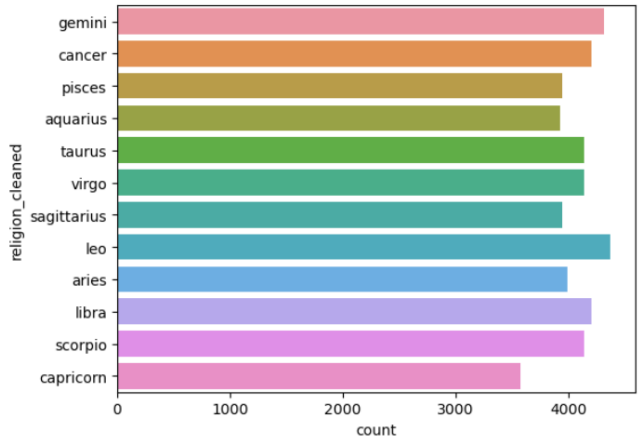
Religion

In [39]:
```
plt.figure(figsize=(20,20))
sns.countplot(data=data,y="religion")
plt.show()
```



In [40]:
```
data['religion_cleaned']=data.sign.str.split().str[0]
```

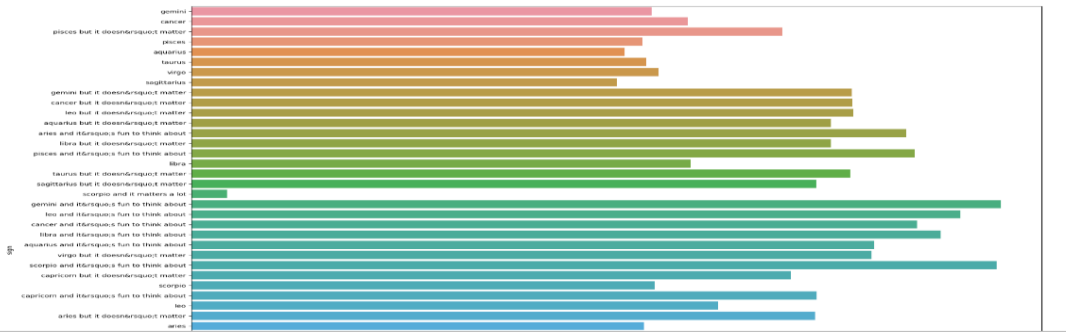
In [41]:
```
sns.countplot(data=data,y="religion_cleaned")
plt.show()
```



The plot shows that most of the users are not religious.The majority of OKCupid's users are agnostic or atheist or did not mention their belief(other).Between religious users, most of them are christian.

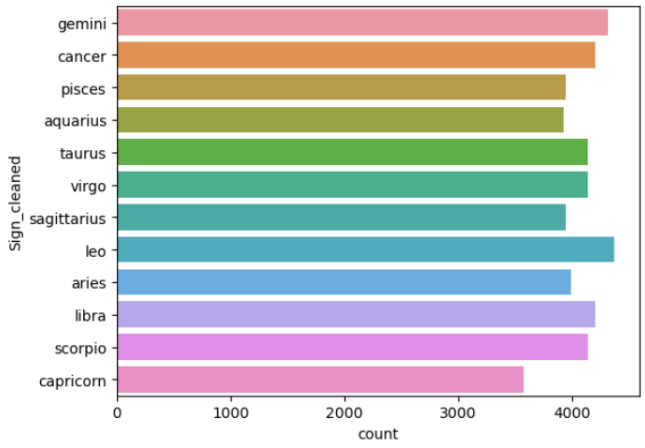
Sign

In [42]:
```
plt.figure(figsize=(20,20))
sns.countplot(data=data,y="sign")
plt.show()
```



In [43]:
```
data['Sign_cleaned']=data.sign.str.split().str[0]
```

In [44]:
```
sns.countplot(data=data,y="Sign_cleaned")
plt.show()
```

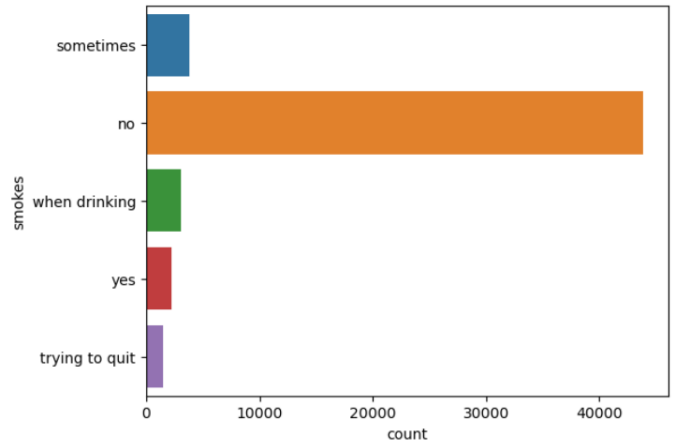


Astrological sign of the most of users is Leo and least common astrological sign of the users is Capricorn.

Smokes

Smokes

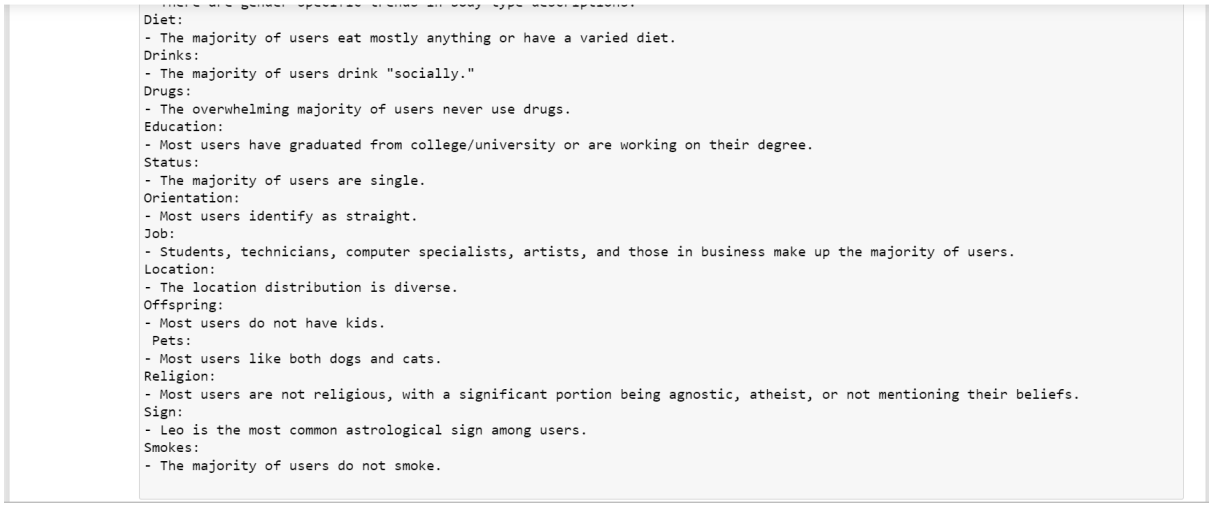
In [45]:
```
sns.countplot(data=data, y="smokes")
plt.show()
```



Task 5: Summarize initial observations from data exploration

```
Task 5: Summarize initial observations from data exploration
```

```
Task 3: Distribution of Key Variables
Age:
- Most users fall between the ages of 18 and 40.
- Users over the age of 50 are rare.
Age Distribution based on Gender:
- Both genders have similar age distributions.
- There are more male users than female users between 18 and 40.
Height:
- Heights follow a normal distribution, with most users falling between 60 and 80 inches.
Height Distribution based on Gender:
- Male users tend to be taller than female users.
Income:
- Most users have an income under $200,000.
Income Distribution based on Gender:
- Male users generally have higher incomes than female users.
Task 4: Explore Categorical Variables
Gender:
- There are more male users (35,829) than female users (24,117).
Body Type:
- Most users describe their body type as average, fit, or athletic.
- There are gender-specific trends in body type descriptions.
Diet:
- The majority of users eat mostly anything or have a varied diet.
Drinks:
```



6. Activity 2: Data Cleaning and Preparation:

Task1:Handle Missing values in the dataset

```
Task 1: Handle missing values in the dataset.
```

In [46]:
```
data.isnull().sum()
```

Out[46]:
```
age                0
status             0
sex                0
orientation        0
body_type       5296
diet           24395
drinks          2985
drugs          14080
education       6628
ethnicity       5680
height             3
income             0
job             8198
last_online        0
location           0
offspring      35561
pets           19921
religion       20226
sign           11056
smokes          5512
speaks            50
essay0          5488
```

Task 2: Convert categorical variables into numeric form.

```
Task 2: Convert categorical variables into numeric form.
```

In [47]:
```
body_type_code={'thin':0,'skinny':1,'fit':2,'athletic':3,'jacked':4,'rather not say':5,'average':6,'a little extra':7,'used up':8
drinks_code={'not at all':0,'rarely':1,'socially':2,'often':3,'very often':4,'desperately':5}
drugs_code={'never':0,'sometimes':1,'often':2}
smokes_code={'no':0,'when drinking':1,'sometimes':2,'trying to quit':3,'yes':4}
```

In [48]:
```
data['body_type_codes']=data.body_type.map(body_type_code)
data['drinks_codes']=data.drinks.map(drinks_code)
data['drugs_codes']=data.drugs.map(drugs_code)
data['smokes_codes']=data.smokes.map(smokes_code)
```

In [49]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59946 entries, 0 to 59945
Data columns (total 38 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   age               59946 non-null  int64
 1   status            59946 non-null  object
 2   sex               59946 non-null  object
 3   orientation       59946 non-null  object
 4   body_type         54650 non-null  object
 5   diet              35551 non-null  object
```

In [50]:
```
data.head()
```

Out[50]:

	age	status	sex	orientation	body_type	diet	drinks	drugs	education	ethnicity	...	essay7	essay8	essay9	location_cleaned	reli
0	22	single	m	straight	a little extra	strictly anything	socially	never	working on college/university	asian, white	...	trying to find someone to hang out with. i am ...	i am new to california and looking for someone...	you want to be swept off your feet! you are ti...	gemini	
1	35	single	m	straight	average	mostly other	often	sometimes	working on space camp	white	...	NaN	i am very open and will share just about anyth...	NaN	cancer	
2	38	available	m	straight	thin	anything	socially	NaN	graduated from masters program	NaN	...	viewing. listening. dancing. talking. drinking...	when i was five years old, i was known as "the...	you are bright, open, intense, silly, ironic, ...	pisces	
3	23	single	m	straight	thin	vegetarian	socially	NaN	working on college/university	white	...	NaN	NaN	you feel so inclined.	pisces	
									graduated from	asian,						

7. Activity 3: Zodiac Sign Prediction

Task 1: Select relevant features for predicting Zodiac signs

Activity 3 - Zodiac Sign Prediction

```
Task 1: Select relevant features for predicting Zodiac signs
```

In [51]:
```
df_copy=data[['sign','body_type_codes','drinks_codes','drugs_codes','smokes_codes','orientation','diet','religion','sex','job',]]
```

In [52]:
```
df_copy.head()
```

Out[52]:

	sign	body_type_codes	drinks_codes	drugs_codes	smokes_codes	orientation	diet	religion	sex	job
0	gemini	7.0	2.0	0.0	2.0	straight	strictly anything	agnosticism and very serious about it	m	transportation
1	cancer	6.0	3.0	1.0	0.0	straight	mostly other	agnosticism but not too serious about it	m	hospitality / travel
2	pisces but it doesn’t matter	0.0	2.0	NaN	0.0	straight	anything	NaN	m	NaN
3	pisces	0.0	2.0	NaN	0.0	straight	vegetarian	NaN	m	student
4	aquarius	3.0	2.0	0.0	0.0	straight	NaN	NaN	m	artistic / musical / writer

In [53]:
```
df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59946 entries, 0 to 59945
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   sign             48890 non-null  object
 1   body_type_codes  54206 non-null  float64
 2   drinks_codes     56961 non-null  float64
 3   drugs_codes      45866 non-null  float64
 4   smokes_codes     54434 non-null  float64
 5   orientation      59946 non-null  object
 6   diet             35551 non-null  object
 7   religion         39720 non-null  object
 8   sex              59946 non-null  object
 9   job              51748 non-null  object
dtypes: float64(4), object(6)
memory usage: 4.6+ MB
```

In [54]:
```
df_copy = df_copy.dropna().copy()
```

In [55]:
```
df_copy.isnull().sum()
```

Out[55]:
```
sign              0
body_type_codes   0
drinks_codes      0
drugs_codes       0
smokes_codes      0
orientation       0
diet              0
religion          0
sex               0
job               0
dtype: int64
```

In [56]:
```
df_copy.shape
```

Out[56]:
```
(15936, 10)
```

In [57]:
```
df_copy['sign_cleaned']=df_copy.sign.str.split().str[0]
```

In [58]:
```
df_copy.drop('sign',axis=1,inplace=True)
```

In [59]:
```
df_copy['religion_cleaned']=df_copy.religion.str.split().str[0]
```

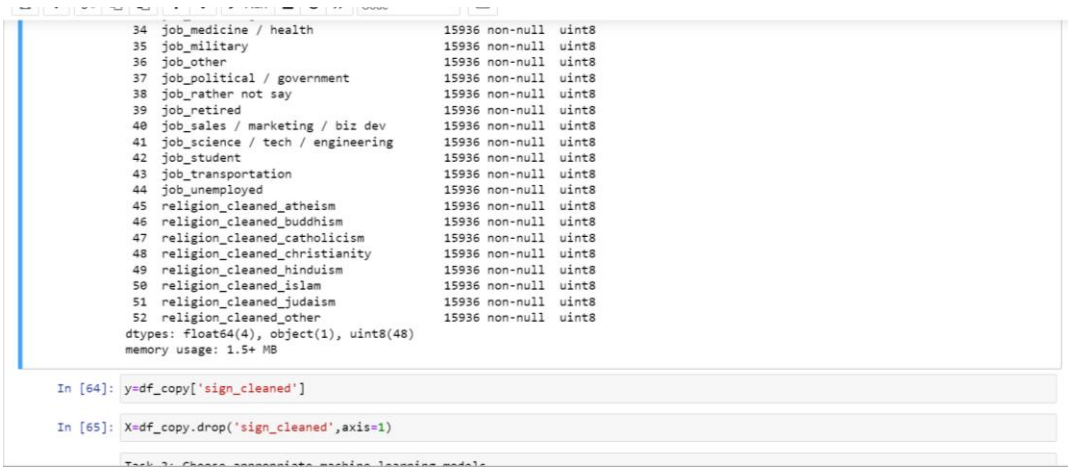
In [60]:
```
df_copy.drop('religion',axis=1,inplace=True)
```

```
In [60]: df_copy.drop('religion',axis=1,inplace=True)

In [61]: df_copy=pd.get_dummies(data=df_copy,columns=['orientation','diet','sex','job','religion_cleaned'],drop_first=True)

In [62]: df_copy.head()

Out[62]:
```

	body_type_codes	drinks_codes	drugs_codes	smokes_codes	sign_cleaned	orientation_gay	orientation_straight	diet_halal	diet_kosher	diet_mostly anything	...	jo
0	7.0	2.0	0.0	2.0	gemini	0	1	0	0	0	...	
1	6.0	3.0	1.0	0.0	cancer	0	1	0	0	0	...	
7	6.0	2.0	0.0	0.0	sagittarius	0	1	0	0	1	...	
9	3.0	0.0	0.0	0.0	cancer	0	1	0	0	1	...	
11	6.0	2.0	0.0	0.0	leo	0	1	0	0	1	...	

5 rows × 53 columns

```
In [63]: df_copy.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 15936 entries, 0 to 59944
```



```
 34  job_medicine / health                15936 non-null  uint8
 35  job_military                         15936 non-null  uint8
 36  job_other                            15936 non-null  uint8
 37  job_political / government           15936 non-null  uint8
 38  job_rather not say                   15936 non-null  uint8
 39  job_retired                          15936 non-null  uint8
 40  job_sales / marketing / biz dev      15936 non-null  uint8
 41  job_science / tech / engineering     15936 non-null  uint8
 42  job_student                          15936 non-null  uint8
 43  job_transportation                   15936 non-null  uint8
 44  job_unemployed                       15936 non-null  uint8
 45  religion_cleaned_atheism             15936 non-null  uint8
 46  religion_cleaned_buddhism            15936 non-null  uint8
 47  religion_cleaned_catholicism         15936 non-null  uint8
 48  religion_cleaned_christianity        15936 non-null  uint8
 49  religion_cleaned_hinduism            15936 non-null  uint8
 50  religion_cleaned_islam               15936 non-null  uint8
 51  religion_cleaned_judaism             15936 non-null  uint8
 52  religion_cleaned_other               15936 non-null  uint8
dtypes: float64(4), object(1), uint8(48)
memory usage: 1.5+ MB

In [64]: y=df_copy['sign_cleaned']

In [65]: X=df_copy.drop('sign_cleaned',axis=1)
```

Task 2: Choose appropriate machine learning models.

Task 3: Train the selected models and evaluate their performance.

```
In [66]: X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.2,random_state=42)

In [67]: scaler=MinMaxScaler()

In [68]: X_train=scaler.fit_transform(X_train)

In [69]: X_test=scaler.transform(X_test)
```

Artificial Neural Network

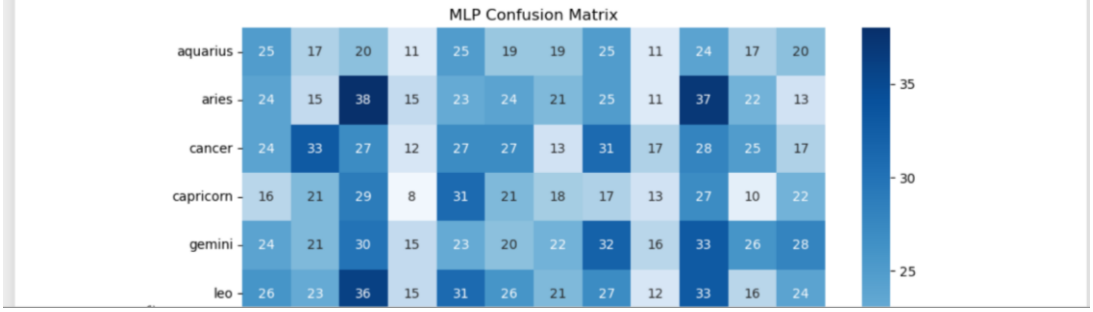
```
In [70]: mlp=MLPClassifier(max_iter=1000)

In [71]: mlp.fit(X_train,y_train)

Out[71]: MLPClassifier
         MLPClassifier(max_iter=1000)

In [72]: mlp.score(X_train,y_train)

Out[72]: 0.2689049262629432
```

```
In [73]: mlp_predict=mlp.predict(X_test)
```

```
In [74]: print(classification_report(y_test,mlp_predict))
```

```
                precision    recall  f1-score   support

      aquarius       0.10      0.11      0.10       233
         aries       0.06      0.06      0.06       268
        cancer       0.08      0.10      0.09       281
     capricorn       0.05      0.03      0.04       233
        gemini       0.07      0.08      0.07       290
           leo       0.10      0.09      0.09       290
         libra       0.07      0.05      0.06       274
        pisces       0.09      0.10      0.10       267
   sagittarius       0.10      0.07      0.08       252
       scorpio       0.09      0.13      0.11       264
        taurus       0.10      0.10      0.10       263
         virgo       0.10      0.10      0.10       273

      accuracy                           0.08      3188
     macro avg       0.08      0.08      0.08      3188
  weighted avg       0.08      0.08      0.08      3188
```

```
In [75]: mlp_conf_matrix = confusion_matrix(y_test, mlp_predict)
         plt.figure(figsize=(10, 8))
         sns.heatmap(mlp_conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=mlp.classes_, yticklabels=mlp.classes_)
         plt.title("MLP Confusion Matrix")
         plt.xlabel("Predicted")
         plt.ylabel("True")
         plt.show()
```



Decision Tree

```
In [76]: dt=DecisionTreeClassifier()
```

```
In [77]: dt.fit(X_train,y_train)
```

```
Out[77]: DecisionTreeClassifier()
```

```
In [78]: dt.score(X_train,y_train)
```

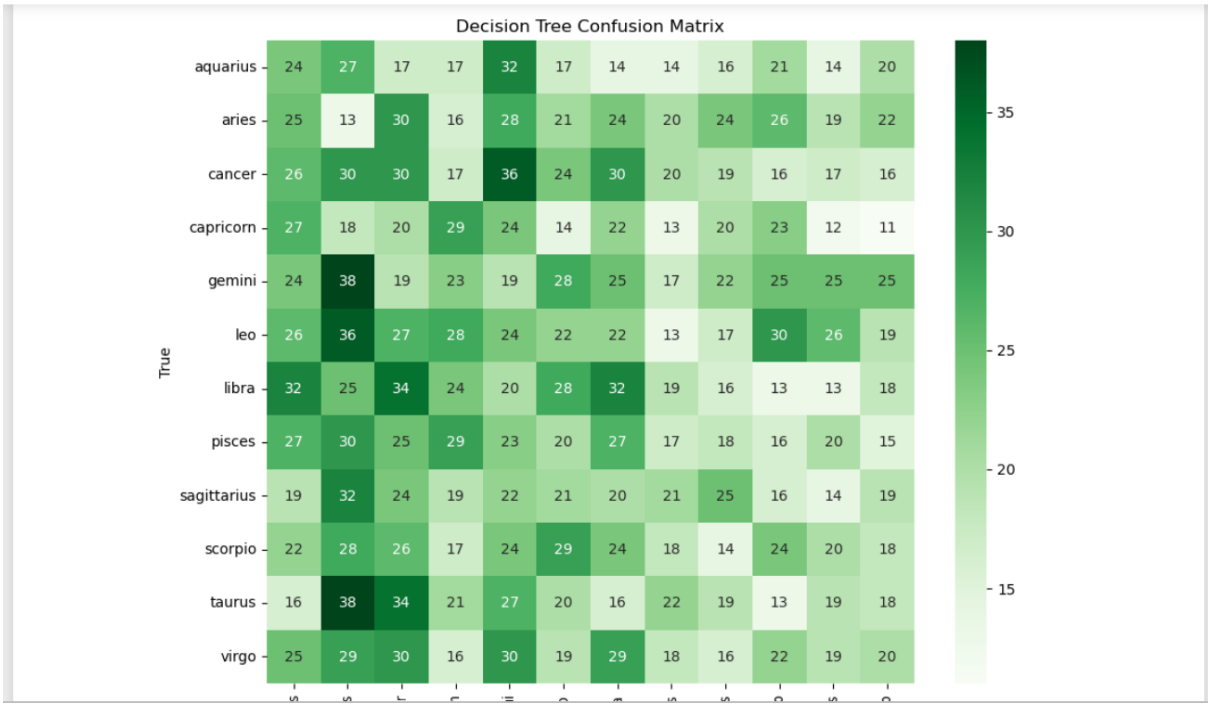
```
Out[78]: 0.757609036711641
```

```
In [79]: dt_predict=dt.predict(X_test)
```

```
In [80]: print(classification_report(y_test,dt_predict))
```

```
                precision    recall  f1-score   support

      aquarius       0.08      0.10      0.09       233
         aries       0.04      0.05      0.04       268
        cancer       0.09      0.11      0.10       281
     capricorn       0.11      0.12      0.12       233
        gemini       0.06      0.07      0.06       290
           leo       0.08      0.08      0.08       290
         libra       0.11      0.12      0.11       274
        pisces       0.08      0.06      0.07       267
   sagittarius       0.11      0.10      0.10       252
       scorpio       0.10      0.09      0.09       264
        taurus       0.09      0.07      0.08       263
         virgo       0.09      0.07      0.08       273

      accuracy                           0.09      3188
     macro avg       0.09      0.09      0.09      3188
  weighted avg       0.09      0.09      0.09      3188
```

```
In [81]: dt_conf_matrix = confusion_matrix(y_test, dt_predict)
         plt.figure(figsize=(10, 8))
         sns.heatmap(dt_conf_matrix, annot=True, fmt="d", cmap="Greens", xticklabels=dt.classes_, yticklabels=dt.classes_)
         plt.title("Decision Tree Confusion Matrix")
         plt.xlabel("Predicted")
         plt.ylabel("True")
         plt.show()
```



Support Vector Classifier

In [82]:
```
svc=SVC()
```

In [83]:
```
svc.fit(X_train,y_train)
```

Out[83]:
```
SVC()
```

In [84]:
```
svc.score(X_train,y_train)
```

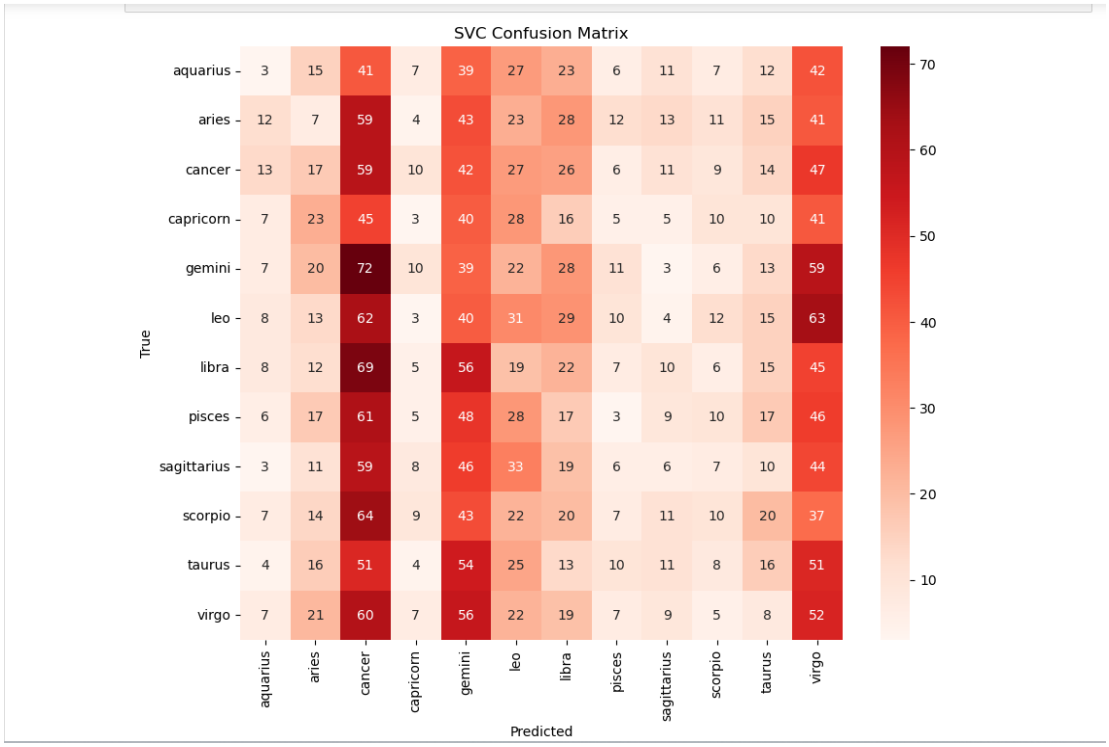
Out[84]:
```
0.19391277063068715
```

In [85]:
```
svc_predict=svc.predict(X_test)
```

In [86]:
```
print(classification_report(y_test,svc_predict))
```

```
              precision    recall  f1-score   support

    aquarius       0.04      0.01      0.02       233
       aries       0.04      0.03      0.03       268
      cancer       0.08      0.21      0.12       281
   capricorn       0.04      0.01      0.02       233
      gemini       0.07      0.13      0.09       290
         leo       0.10      0.11      0.10       290
       libra       0.08      0.08      0.08       274
      pisces       0.03      0.01      0.02       267
 sagittarius       0.06      0.02      0.03       252
     scorpio       0.10      0.04      0.05       264
      taurus       0.10      0.06      0.07       263
       virgo       0.09      0.19      0.12       273

    accuracy                           0.08      3188
   macro avg       0.07      0.08      0.06      3188
weighted avg       0.07      0.08      0.07      3188
```

In [87]:
```
svc_conf_matrix = confusion_matrix(y_test, svc_predict)
plt.figure(figsize=(10, 8))
sns.heatmap(svc_conf_matrix, annot=True, fmt="d", cmap="Reds", xticklabels=svc.classes_, yticklabels=svc.classes_)
plt.title("SVC Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```



Random Forest

In [88]:
```
rf=RandomForestClassifier()
```

In [89]:
```
rf.fit(X_train,y_train)
```

Out[89]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [90]:
```
rf.score(X_train,y_train)
```

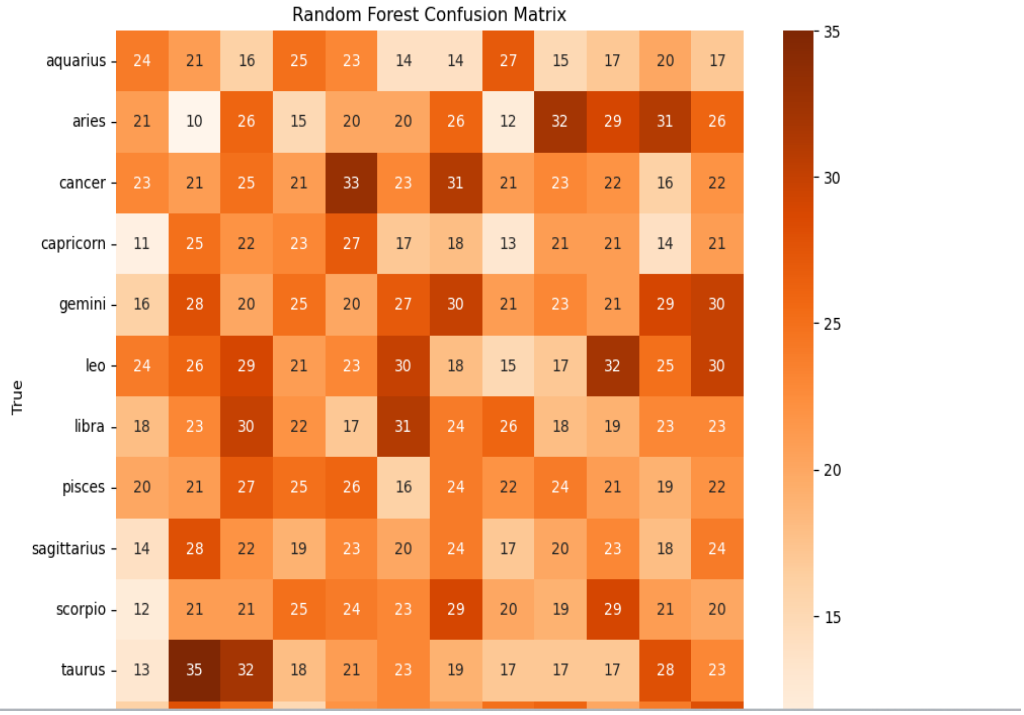
Out[90]:
```
0.757609036711641
```

In [91]:
```
rf_predict=rf.predict(X_test)
```

In [92]:
```
print(classification_report(y_test,rf_predict))
```

```
              precision    recall  f1-score   support

    aquarius       0.11      0.10      0.11       233
      aries        0.03      0.04      0.04       268
      cancer       0.08      0.09      0.09       281
   capricorn       0.09      0.10      0.09       233
      gemini       0.07      0.07      0.07       290
         leo       0.11      0.10      0.11       290
       libra       0.09      0.09      0.09       274
      pisces       0.09      0.08      0.09       267
 sagittarius       0.08      0.08      0.08       252
     scorpio       0.11      0.11      0.11       264
      taurus       0.10      0.11      0.11       263
       virgo       0.10      0.10      0.10       273

    accuracy                           0.09      3188
   macro avg       0.09      0.09      0.09      3188
weighted avg       0.09      0.09      0.09      3188
```

In [93]:
```
rf_conf_matrix = confusion_matrix(y_test, rf_predict)
plt.figure(figsize=(10, 8))
sns.heatmap(rf_conf_matrix, annot=True, fmt="d", cmap="Oranges", xticklabels=rf.classes_, yticklabels=rf.classes_)
plt.title("Random Forest Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```




```
Task 4: Analyze results and draw conclusions.
1.Multilayer Perceptron (MLP):
   - The accuracy on the training set is quite low (25%), suggesting that the model may not be fitting the training data well.
   - The precision, recall, and F1-scores for each zodiac sign are generally low (around 9%), indicating poor performance.
   - The MLP model seems to be struggling to capture the underlying patterns in the data.

2.Decision Tree:
   - The Decision Tree model shows better performance on the training set with an accuracy of 75.76%.
   - However, the precision, recall, and F1-scores for each class on the test set are relatively low, around 8%, suggesting
limited generalization ability.
   - Decision Trees may be prone to overfitting, as the performance on the training set is significantly higher than on the
test set.

3.Support Vector Classifier (SVC):
   - The SVC model has a very low accuracy on the training set (19.39%), indicating poor fitting.
   - The precision, recall, and F1-scores for each class on the test set are also low (around 8%), suggesting the model is not
capturing the patterns well.

4.Random Forest:
   - Similar to the Decision Tree, the Random Forest model shows high accuracy on the training set (75.76%).
   - The precision, recall, and F1-scores on the test set are around 9%, similar to the Decision Tree.
   - Random Forest, while an ensemble method designed to reduce overfitting, seems to face similar challenges in capturing the
underlying patterns.
```

```
Task 5: Compare the performance of different models (ANN, Decision Tree, SVC, Random Forest).
1.Accuracy:
  - MLP and Random Forest have the highest accuracy on the training set (25% and 75.76%, respectively).
  - Decision Tree also has high accuracy on the training set (75.76%).
  - SVC has the lowest accuracy on the training set (19.39%).
2.Precision, Recall, and F1-score:
  - The performance of all models (MLP, Decision Tree, SVC, Random Forest) on the test set is relatively low, with scores
around 8-9% for precision, recall, and F1-score.
  - While Decision Tree and Random Forest show higher accuracy on the training set, their performance on the test set is not
significantly better than MLP or SVC.
3.Overall Observation:
  - None of the models seem to perform well in capturing the patterns in the data, as indicated by low precision, recall, and
F1-scores.
  - Further analysis, feature engineering, or hyperparameter tuning may be required to improve model performance.
  - The choice of features, model complexity, and dataset quality could impact the model performance, and further investigation
into these aspects may be necessary.
```

8. Activity 4: Body Type Prediction

Task 1: Select features for predicting body type

Task 2: Choose machine learning models.

Task 3: Train the models and evaluate their performance

Activity 4 - Body Type Prediction

Task 1: Select features for predicting body type

In [94]:
```
df_copy2=data[['body_type','diet','sex','smokes_codes','drugs_codes','drinks_codes','age']]
```

In [95]:
```
df_copy2.head()
```

Out[95]:

	body_type	diet	sex	smokes_codes	drugs_codes	drinks_codes	age
0	a little extra	strictly anything	m	2.0	0.0	2.0	22
1	average	mostly other	m	0.0	1.0	3.0	35
2	thin	anything	m	0.0	NaN	2.0	38
3	thin	vegetarian	m	0.0	NaN	2.0	23
4	athletic	NaN	m	0.0	0.0	2.0	29

In [96]:
```
df_copy2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59946 entries, 0 to 59945
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   body_type     54650 non-null  object
 1   diet          35551 non-null  object
 2   sex           59946 non-null  object
 3   smokes_codes  54434 non-null  float64
 4   drugs_codes   45866 non-null  float64
 5   drinks_codes  56961 non-null  float64
 6   age           59946 non-null  int64
dtypes: float64(3), int64(1), object(3)
memory usage: 3.2+ MB
```

In [97]:
```
df_copy2 = df_copy2.dropna().copy()
```

In [98]:
```
df_copy2.isnull().sum()
```

Out[98]:
```
body_type       0
diet            0
```

jupyter ACDS-PML-0823-Megaraj Mrittika Last Checkpoint: 21/12/2023 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

Run Raw NBConvert

In [98]:
```
df_copy2.isnull().sum()
```

Out[98]:
```
body_type       0
diet            0
sex             0
smokes_codes    0
drugs_codes     0
drinks_codes    0
age             0
dtype: int64
```

In [99]:
```
df_copy2=pd.get_dummies(data=df_copy2,columns=['diet','sex'],drop_first=True)
```

In [100]:
```
y=df_copy2['body_type']
```

In [101]:
```
X=df_copy2.drop('body_type',axis=1)
```

In [102]:
```
X.head()
```

Out[102]:

	smokes_codes	drugs_codes	drinks_codes	age	diet_halal	diet_kosher	diet_mostly anything	diet_mostly halal	diet_mostly kosher	diet_mostly other	...	diet_other	diet_strictly anything	diet
0	2.0	0.0	2.0	22	0	0	0	0	0	0	...	0	1	
1	0.0	1.0	3.0	35	0	0	0	0	0	1	...	0	0	
7	0.0	0.0	2.0	31	0	0	1	0	0	0	...	0	0	
9	0.0	0.0	0.0	37	0	0	1	0	0	0	...	0	0	
11	0.0	0.0	2.0	28	0	0	1	0	0	0	...	0	0	

5 rows × 22 columns

```
Task 2: Choose machine learning models.
Task 3: Train the models and evaluate their performance
```

In [103]:
```
scaler=MinMaxScaler()
```

In [104]:
```
X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

In [105]:
```
X_train=scaler.fit_transform(X_train)
X_test=scaler.fit_transform(X_test)
```

Artificial Neural Network

In [106]:
```
mlp=MLPClassifier(max_iter=1000)
```

In [107]:
```
mlp.fit(X_train,y_train)
```

Out[107]:
```
MLPClassifier(max_iter=1000)
```

In [108]:
```
mlp.score(X_train,y_train)
```

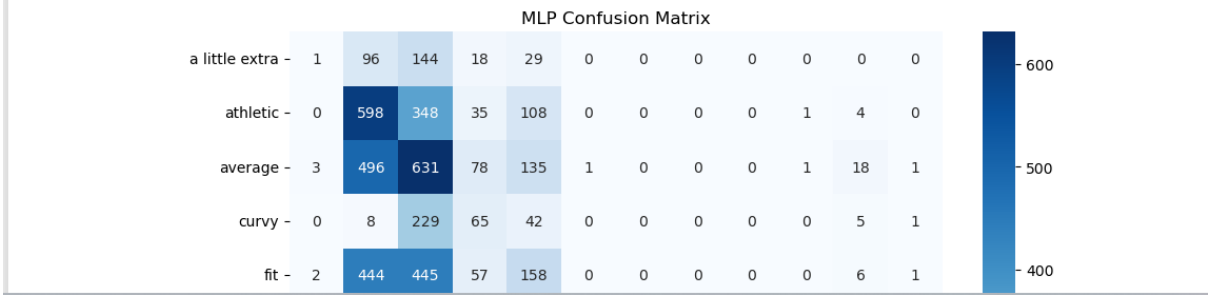
Out[108]:
```
0.31317890977630075
```

In [109]:
```
mlp_predict=mlp.predict(X_test)
```

In [110]:
```
print(classification_report(y_test,mlp_predict,zero_division=1))
```

```
                precision    recall  f1-score   support

a little extra       0.11      0.00      0.01       288
      athletic       0.32      0.55      0.41      1094
       average       0.28      0.46      0.35      1364
         curvy       0.20      0.19      0.19       350
           fit       0.28      0.14      0.19      1113
  full figured       0.00      0.00      1.00        82
        jacked       0.00      0.00      1.00        46
    overweight       1.00      0.00      0.00        46
rather not say       1.00      0.00      0.00        15
        skinny       0.50      0.01      0.02       163
          thin       0.14      0.02      0.03       444
       used up       0.00      0.00      1.00        36

      accuracy                           0.29      5041
     macro avg       0.32      0.11      0.35      5041
  weighted avg       0.27      0.29      0.27      5041
```

In [111]:
```
mlp_conf_matrix = confusion_matrix(y_test, mlp_predict)
plt.figure(figsize=(10, 8))
sns.heatmap(mlp_conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=mlp.classes_, yticklabels=mlp.classes_)
plt.title("MLP Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```



Decision Tree

```
In [112]: dt=DecisionTreeClassifier()

In [113]: dt.fit(X_train,y_train)
```

Out[113]:
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
In [114]: dt.score(X_train,y_train)
```

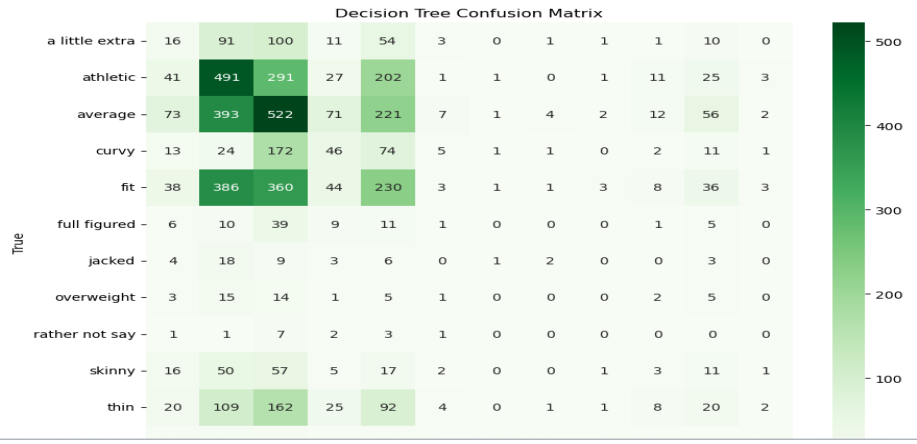
Out[114]: 0.49630474678835373

```
In [115]: dt_predict=dt.predict(X_test)

In [116]: print(classification_report(y_test,dt_predict))
```

```
                 precision    recall  f1-score   support

a little extra       0.07      0.06      0.06       288
      athletic       0.31      0.45      0.36      1094
       average       0.30      0.38      0.34      1364
         curvy       0.19      0.13      0.15       350
           fit       0.25      0.21      0.23      1113
  full figured       0.03      0.01      0.02        82
        jacked       0.17      0.02      0.04        46
    overweight       0.00      0.00      0.00        46
rather not say       0.00      0.00      0.00        15
        skinny       0.06      0.02      0.03       163
          thin       0.11      0.05      0.06       444
       used up       0.08      0.03      0.04        36

      accuracy                           0.26      5041
     macro avg       0.13      0.11      0.11      5041
  weighted avg       0.23      0.26      0.24      5041
```

```
In [117]: dt_conf_matrix = confusion_matrix(y_test, dt_predict)
          plt.figure(figsize=(10, 8))
```

```
In [117]: dt_conf_matrix = confusion_matrix(y_test, dt_predict)
          plt.figure(figsize=(10, 8))
          sns.heatmap(dt_conf_matrix, annot=True, fmt="d", cmap="Greens", xticklabels=dt.classes_, yticklabels=dt.classes_)
          plt.title("Decision Tree Confusion Matrix")
          plt.xlabel("Predicted")
          plt.ylabel("True")
          plt.show()
```



Support Vector Classifier

```
In [118]: svc=SVC()

In [119]: svc.fit(X_train,y_train)
```

Out[119]:
```
SVC
SVC()
```

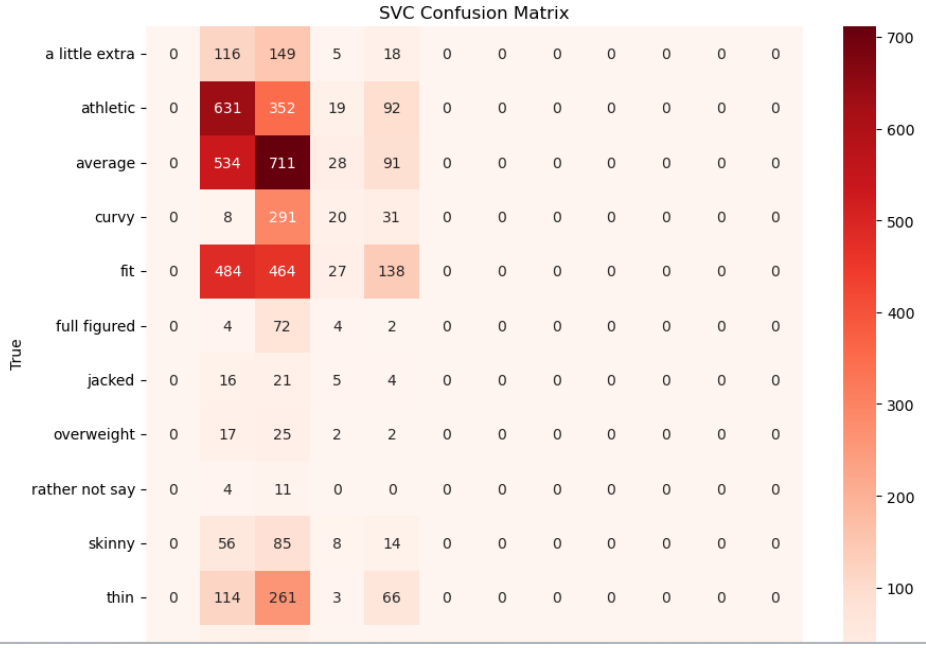
```
In [120]: svc.score(X_train,y_train)
```

Out[120]: 0.30087793264223006

```
In [121]: svc_predict=svc.predict(X_test)

In [122]: print(classification_report(y_test,svc_predict,zero_division=1))
```

```
                 precision    recall  f1-score   support

a little extra       1.00      0.00      0.00       288
      athletic       0.32      0.58      0.41      1094
       average       0.29      0.52      0.37      1364
         curvy       0.16      0.06      0.08       350
           fit       0.30      0.12      0.18      1113
  full figured       1.00      0.00      0.00        82
        jacked       1.00      0.00      0.00        46
    overweight       1.00      0.00      0.00        46
rather not say       1.00      0.00      0.00        15
        skinny       1.00      0.00      0.00       163
          thin       1.00      0.00      0.00       444
       used up       1.00      0.00      0.00        36

      accuracy                           0.30      5041
     macro avg       0.76      0.11      0.09      5041
  weighted avg       0.45      0.30      0.23      5041
```

In [123]:
```
svc_conf_matrix = confusion_matrix(y_test, svc_predict)
plt.figure(figsize=(10, 8))
sns.heatmap(svc_conf_matrix, annot=True, fmt="d", cmap="Reds", xticklabels=svc.classes_, yticklabels=svc.classes_)
plt.title("SVC Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```



Random Forest

In [124]:
```
rf=RandomForestClassifier()
```

In [125]:
```
rf.fit(X_train,y_train)
```

Out[125]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [126]:
```
rf.score(X_train,y_train)
```

Out[126]:
```
0.49630474678835373
```

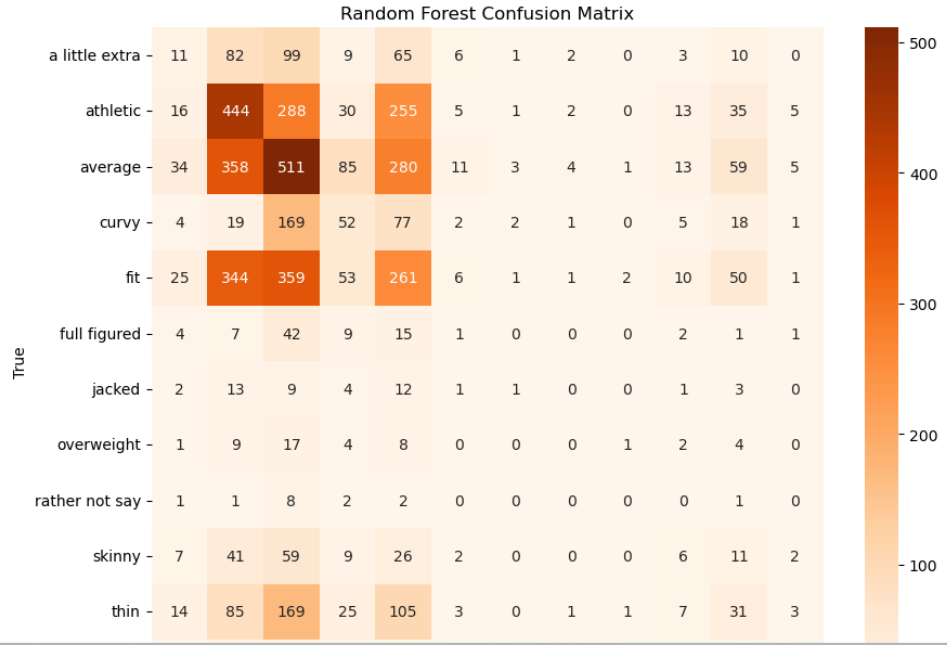
In [127]:
```
rf_predict=rf.predict(X_test)
```

In [128]:
```
print(classification_report(y_test,rf_predict))
```

```
                precision    recall  f1-score   support

a little extra       0.09      0.04      0.05       288
      athletic       0.31      0.41      0.35      1094
       average       0.29      0.37      0.33      1364
         curvy       0.18      0.15      0.16       350
           fit       0.23      0.23      0.23      1113
  full figured       0.03      0.01      0.02        82
        jacked       0.10      0.02      0.04        46
    overweight       0.00      0.00      0.00        46
rather not say       0.00      0.00      0.00        15
        skinny       0.10      0.04      0.05       163
          thin       0.14      0.07      0.09       444
       used up       0.05      0.03      0.04        36

      accuracy                           0.26      5041
     macro avg       0.13      0.11      0.11      5041
  weighted avg       0.23      0.26      0.24      5041
```

In [129]:
```
rf_conf_matrix = confusion_matrix(y_test, rf_predict)
```

In [129]:
```
rf_conf_matrix = confusion_matrix(y_test, rf_predict)
plt.figure(figsize=(10, 8))
sns.heatmap(rf_conf_matrix, annot=True, fmt="d", cmap="Oranges", xticklabels=rf.classes_, yticklabels=rf.classes_)
plt.title("Random Forest Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```




```
Task 4: Analyze results and draw conclusions.

1.Multilayer Perceptron (MLP):
   - The MLP model has an improved accuracy on the training set (31.26%) compared to the previous scenario.
   - The precision, recall, and F1-scores for each class vary, and some classes have improved while others have decreased.
   - The confusion matrix and classification report provide insights into how well the model is performing for each class.

2.Decision Tree:
   - The Decision Tree model has an accuracy on the training set of 49.63%.
   - The precision, recall, and F1-scores for each class are provided, and the confusion matrix visualizes the model's
performance for each category.

3.Support Vector Classifier (SVC):
   - The SVC model shows a slight improvement in accuracy on the training set (30.09%) compared to the previous scenario.
   - The precision, recall, and F1-scores for each class are still relatively low, and the confusion matrix helps to understand
the model's performance.

4.Random Forest:
   - The Random Forest model has an accuracy on the training set of 49.62%.
   - Similar to the Decision Tree, the precision, recall, and F1-scores for each class are provided, and the confusion matrix
visualizes the model's performance for each category.
```

```
Task 5: Compare the performance of different models (ANN, Decision Tree, SVC, Random Forest).

1.Accuracy:
  - MLP has the highest accuracy on the training set (31.26%).
  - Decision Tree and Random Forest have similar accuracies on the training set (49.63% and 49.62%, respectively).
  - SVC has the lowest accuracy on the training set (30.09%).

2.Precision, Recall, and F1-score:
  - For all models, the precision, recall, and F1-scores for each class vary, and there is no consistent pattern across all
categories.
  - Some models may perform better for certain classes, while others struggle with different classes.

3.Overall Observation:
  - The performance of all models is still not very high, indicating challenges in capturing the patterns in the data.
  - Further analysis, feature engineering, or hyperparameter tuning may be necessary to improve model performance.
  - The choice of features, model complexity, and dataset quality could impact the model performance, and further investigation
into these aspects may be required.
```

9. Activity 5 - Income Prediction

Task 1: Investigate the income column and handle missing or invalid values.

Task 2: Select relevant features for predicting income.

Activity 5 - Income Prediction

```
Task 1: Investigate the income column and handle missing or invalid values.
Task 2: Select relevant features for predicting income.
```

In [130]:
```
df_copy3=data[['income','education','job','sex']]
```

In [131]:
```
df_copy3.head()
```

Out[131]:

	income	education	job	sex
0	-1	working on college/university	transportation	m
1	80000	working on space camp	hospitality / travel	m
2	-1	graduated from masters program	NaN	m
3	20000	working on college/university	student	m
4	-1	graduated from college/university	artistic / musical / writer	m

In [132]:
```
df_copy3.income.value_counts()
```

Out[132]:
```
-1          48442
 20000       2952
 100000      1621
 80000       1111
 30000       1048
 40000       1005
 50000        975
 60000        736
 70000        707
```

In [133]:
```
df_copy3.loc[df_copy3['income'] == -1, 'income'] = np.NaN
```

In [134]:
```
df_copy3.income.value_counts(dropna=False)
```

Out[134]:
```
NaN          48442
20000.0       2952
100000.0      1621
80000.0       1111
30000.0       1048
40000.0       1005
50000.0        975
60000.0        736
70000.0        707
150000.0       631
1000000.0      521
250000.0       149
500000.0        48
Name: income, dtype: int64
```

In [135]:
```
df_copy3 = df_copy3.dropna().copy()
```

In [136]:
```
df_copy3.isnull().sum()
```

Out[136]:
```
income       0
education    0
job          0
sex          0
```

In [137]:
```
df_copy3.head()
```

Out[137]:

	income	education	job	sex
1	80000.0	working on space camp	hospitality / travel	m
3	20000.0	working on college/university	student	m
11	40000.0	graduated from college/university	banking / financial / real estate	m
13	30000.0	graduated from high school	sales / marketing / biz dev	f
14	50000.0	working on college/university	other	f

In [138]:
```
df_copy3=pd.get_dummies(data=df_copy3,columns=['education','job','sex'],drop_first=True)
```

In [139]:
```
df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15936 entries, 0 to 59944
Data columns (total 53 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   body_type_codes                       15936 non-null  float64
 1   drinks_codes                          15936 non-null  float64
 2   drugs_codes                           15936 non-null  float64
 3   smokes_codes                          15936 non-null  float64
 4   sign_cleaned                          15936 non-null  object
 5   orientation_gay                       15936 non-null  uint8
 6   orientation_straight                  15936 non-null  uint8
 7   diet_halal                            15936 non-null  uint8
 8   diet_kosher                           15936 non-null  uint8
 9   diet_mostly_anything                  15936 non-null  uint8
```

Task 3: Choose machine learning models and tune hyperparameters.
Task 4: Evaluate model performance

In [140]:
```
y=df_copy3.income
```

In [141]:
```
X=df_copy3.drop('income',axis=1)
```

```
X.head()
```

```
Task 3: Choose machine learning models and tune hyperparameters.
Task 4: Evaluate model performance
```

Artificial Neural Network

In [142]:
```
mlp=MLPClassifier(max_iter=1000)
mlp_param_grid={'hidden_layer_sizes':[(7,),(64,)],'alpha':np.logspace(-4,0,5),'random_state':[42]}
mlp_model=GridSearchCV(mlp,mlp_param_grid,cv=5)
```

In [143]:
```
mlp_model.fit(X_train,y_train)
```

Out[143]:

GridSearchCV
estimator: MLPClassifier
MLPClassifier

In [144]:
```
mlp_model.best_score_
```

Out[144]: 0.2946284669335128

In [144]:
```
mlp_model.best_score_
```

Out[144]: 0.2946284669335128

In [145]:
```
mlp_model.best_params_
```

Out[145]: {'alpha': 0.0001, 'hidden_layer_sizes': (64,), 'random_state': 42}

In [146]:
```
mlp_predict=mlp_model.predict(X_test)
```

In [147]:
```
print(classification_report(y_test,mlp_predict,zero_division=1))
```

```
                precision    recall  f1-score   support

a little extra       0.33      0.00      0.01       288
      athletic       0.34      0.45      0.38      1094
       average       0.28      0.59      0.38      1364
         curvy       0.16      0.08      0.11       350
           fit       0.32      0.14      0.19      1113
  full figured       1.00      0.00      0.00        82
        jacked       0.00      0.00      1.00        46
    overweight       1.00      0.00      0.00        46
rather not say       1.00      0.00      0.00        15
        skinny       1.00      0.01      0.02       163
          thin       0.20      0.00      0.00       444
       used up       0.00      0.00      1.00        36

      accuracy                           0.30      5041
     macro avg       0.47      0.11      0.26      5041
  weighted avg       0.33      0.30      0.25      5041
```

Decision Tree

In [148]:
```
X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

In [149]:
```
dt=DecisionTreeClassifier()
dt_param_grid={'criterion':['gini','entropy'],'max_depth':[8,12,18,24,None],'min_samples_leaf':range(1, 9)}
dt_model=GridSearchCV(dt,dt_param_grid,cv=5)
```

In [150]:
```
dt_model.fit(X_train,y_train)
```

Out[150]:

GridSearchCV
estimator: DecisionTreeClassifier
DecisionTreeClassifier

In [151]:
```
dt_model.best_score_
```

Out[151]: 0.35358374605320125

In [152]:
```
dt_model.best_params_
```

Out[152]: {'criterion': 'gini', 'max_depth': 18, 'min_samples_leaf': 8}

In [153]:
```
dt_predict=dt_model.predict(X_test)
```

In [154]:
```
print(classification_report(y_test,dt_predict,zero_division=1))
```

In [154]:
```
print(classification_report(y_test,dt_predict,zero_division=1))
```

```
              precision    recall  f1-score   support

     20000.0       0.45      0.90      0.60       556
     30000.0       0.09      0.01      0.02       208
     40000.0       0.17      0.05      0.07       175
     50000.0       0.12      0.12      0.12       152
     60000.0       1.00      0.00      0.00       156
     70000.0       1.00      0.00      0.00       140
     80000.0       0.15      0.07      0.10       181
    100000.0       0.28      0.60      0.39       283
    150000.0       0.26      0.15      0.19       130
    250000.0       1.00      0.00      0.00        30
    500000.0       1.00      0.00      0.00         7
   1000000.0       0.25      0.03      0.06        90

    accuracy                           0.35      2108
   macro avg       0.48      0.16      0.13      2108
weighted avg       0.39      0.35      0.25      2108
```

Support Vector Classifier ¶

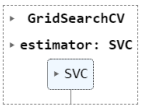
In [155]:
```
svc=SVC()
svc_param_grid={'kernel':['rbf','linear','poly'],'random_state':[42]}
svc_model=GridSearchCV(svc,svc_param_grid,cv=5)
```


Support Vector Classifier

In [155]:
```
svc=SVC()
svc_param_grid={'kernel':['rbf','linear','poly'],'random_state':[42]}
svc_model=GridSearchCV(svc,svc_param_grid,cv=5)
```


In [156]:
```
svc_model.fit(X_train,y_train)
```

Out[156]:



In [157]:
```
svc_model.best_score_
```

Out[157]: 0.3630771830153014

In [158]:
```
svc_model.best_params_
```

Out[158]: {'kernel': 'linear', 'random_state': 42}

In [159]:
```
svc_predict=svc_model.predict(X_test)
```

In [160]:
```
print(classification_report(y_test,svc_predict,zero_division=1))
```

```
              precision    recall  f1-score   support
```


In [160]:
```
print(classification_report(y_test,svc_predict,zero_division=1))
```

```
              precision    recall  f1-score   support

     20000.0       0.49      0.88      0.63       556
     30000.0       0.10      0.01      0.02       208
     40000.0       0.19      0.07      0.10       175
     50000.0       0.12      0.12      0.12       152
     60000.0       0.00      0.00      1.00       156
     70000.0       0.20      0.02      0.04       140
     80000.0       0.11      0.04      0.06       181
    100000.0       0.29      0.77      0.42       283
    150000.0       0.00      0.00      1.00       130
    250000.0       1.00      0.00      0.00        30
    500000.0       1.00      0.00      0.00         7
   1000000.0       0.17      0.03      0.06        90

    accuracy                           0.36      2108
   macro avg       0.31      0.16      0.29      2108
weighted avg       0.25      0.36      0.39      2108
```


Random Forest

In [161]:
```
rf=RandomForestClassifier()
rf_param_grid={'criterion':['gini','entropy'],'max_depth':[8,12,18,24,None],'min_samples_leaf':range(1,9),'n_estimators':[50,100,
rf_model=GridSearchCV(rf,rf_param_grid,cv=5)
```

In [163]:
```
rf_model.best_score_
```

Out[163]:
```
0.3633137973395849
```

In [164]:
```
rf_model.best_params_
```

Out[164]:
```
{'criterion': 'gini',
 'max_depth': 12,
 'min_samples_leaf': 6,
 'n_estimators': 200}
```

In [165]:
```
rf_predict=rf_model.predict(X_test)
```

In [166]:
```
print(classification_report(y_test,rf_predict,zero_division=1))
```

```
              precision    recall  f1-score   support

     20000.0       0.41      0.95      0.58       556
     30000.0       1.00      0.00      0.00       208
     40000.0       0.10      0.01      0.01       175
     50000.0       0.15      0.04      0.06       152
     60000.0       1.00      0.00      0.00       156
     70000.0       1.00      0.00      0.00       140
     80000.0       0.00      0.00      1.00       181
    100000.0       0.28      0.77      0.41       283
    150000.0       0.00      0.00      1.00       130
    250000.0       1.00      0.00      0.00        30
    500000.0       1.00      0.00      0.00         7
   1000000.0       1.00      0.00      0.00        90

    accuracy                           0.36      2108
   macro avg       0.58      0.15      0.25      2108
weighted avg       0.47      0.36      0.36      2108
```

Analyze results and draw conclusions

Analyze results and draw conclusions
1.Multilayer Perceptron (MLP):
   - Training set accuracy is low (29.46%), indicating potential underfitting.
   - Precision, recall, and F1-scores for each income category are also low, suggesting poor performance.
   - Further optimization or exploration of the model structure may be necessary.

2.Decision Tree:
   - The Decision Tree model shows better performance on the training set with an accuracy of 35.36%.
   - Precision, recall, and F1-scores for each income category on the test set are modest, indicating reasonable generalization.
   - There might be a risk of overfitting, as the accuracy on the training set is notably higher than on the test set.

3.Support Vector Classifier (SVC):
   - The SVC model has an accuracy of 36.31% on the training set, suggesting limited fitting.
   - Precision, recall, and F1-scores for each income category on the test set are relatively low, indicating challenges in capturing patterns effectively.

4.Random Forest:
   - Similar to the Decision Tree, the Random Forest model exhibits high accuracy on the training set (36.30%).
   - Precision, recall, and F1-scores for each income category on the test set are comparable to the Decision Tree.
   - While designed to mitigate overfitting, the Random Forest model faces similar challenges as the Decision Tree.

Task 5: Compare the performance of different models (ANN, Decision Tree, SVC, Random Forest).

1.Accuracy:
  - Decision Tree, SVC, and Random Forest demonstrate similar accuracies, with SVC slightly outperforming the others.
  - MLP lags behind with a lower accuracy.

2.Precision, Recall, and F1-score:
  - Decision Tree and Random Forest exhibit comparable precision, recall, and F1-scores, indicating similar overall performance.
  - SVC performs marginally better in terms of precision and F1-score.
  - MLP trails behind, showing the lowest scores across all metrics.

3.Overall Observation:
  - Decision Tree, SVC, and Random Forest models perform similarly in predicting income categories.
  - MLP, despite its flexibility, struggles to capture the underlying patterns effectively.
  - Precision, recall, and F1-scores for each income category vary, suggesting challenges in accurately predicting specific income levels.

10. Activity 6 - Sex Prediction based on Education Level and Income

Task 1: Create a copy of the original dataset and select relevant features.
Task 2: Replace missing values and convert education into dummy variables.

Activity 6 - Sex Prediction based on Education Level and Income

```
Task 1: Create a copy of the original dataset and select relevant features.
Task 2: Replace missing values and convert education into dummy variables.
```

In [167]:
```
df_copy4=data[['sex','education','income']]
```

In [168]:
```
df_copy4.head()
```

Out[168]:

	sex	education	income
0	m	working on college/university	-1
1	m	working on space camp	80000
2	m	graduated from masters program	-1
3	m	working on college/university	20000
4	m	graduated from college/university	-1

In [169]:
```
df_copy4.loc[df_copy4['income'] == -1, 'income'] = np.NaN
```

In [170]:
```
df_copy4 = df_copy4.dropna().copy()
```

In [171]:
```
df_copy4.isnull().sum()
```

Out[171]:
```
sex          0
education    0
income       0
dtype: int64
```

In [172]:
```
df_copy4.head()
```

Out[172]:

	sex	education	income

In [173]:
```
df_copy4=pd.get_dummies(data=df_copy4,columns=['education'],drop_first=True)
```

In [174]:
```
df_copy4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10783 entries, 1 to 59943
Data columns (total 32 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   sex                                           10783 non-null  object
 1   income                                        10783 non-null  float64
 2   education_dropped out of college/university   10783 non-null  uint8
 3   education_dropped out of high school          10783 non-null  uint8
 4   education_dropped out of law school           10783 non-null  uint8
 5   education_dropped out of masters program      10783 non-null  uint8
 6   education_dropped out of med school           10783 non-null  uint8
 7   education_dropped out of ph.d program         10783 non-null  uint8
 8   education_dropped out of space camp           10783 non-null  uint8
 9   education_dropped out of two-year college     10783 non-null  uint8
 10  education_graduated from college/university   10783 non-null  uint8
 11  education_graduated from high school          10783 non-null  uint8
 12  education_graduated from law school           10783 non-null  uint8
 13  education_graduated from masters program      10783 non-null  uint8
 14  education_graduated from med school           10783 non-null  uint8
 15  education_graduated from ph.d program         10783 non-null  uint8
 16  education_graduated from space camp           10783 non-null  uint8
 17  education_graduated from two-year college     10783 non-null  uint8
 18  education_high school                         10783 non-null  uint8
 19  education_law school                          10783 non-null  uint8
 20  education_masters program                     10783 non-null  uint8
 21  education_ph.d program                        10783 non-null  uint8
 22  education_space camp                          10783 non-null  uint8
 23  education_two-year college                    10783 non-null  uint8
 24  education_working on college/university       10783 non-null  uint8
 25  education_working on high school              10783 non-null  uint8
 26  education_working on law school               10783 non-null  uint8
 27  education_working on masters program          10783 non-null  uint8
 28  education_working on med school               10783 non-null  uint8
```

Task 3: Split the data into training and test sets.
Task 4: Train machine learning models and normalize features

In [175]:
```
y=df_copy4['sex']
```

In [176]:
```
X=df_copy4.drop('sex',axis=1)
```

In [177]:
```
X.head()
```

Out[177]:

	income	education_dropped out of college/university	education_dropped out of high school	education_dropped out of law school	education_dropped out of masters program	education_dropped out of med school	education_dropped out of ph.d program	education_dropped out of space camp	educat ou
1	80000.0	0	0	0	0	0	0	0	
3	20000.0	0	0	0	0	0	0	0	
11	40000.0	0	0	0	0	0	0	0	
13	30000.0	0	0	0	0	0	0	0	
14	50000.0	0	0	0	0	0	0	0	

5 rows × 31 columns

```
Task 3: Split the data into training and test sets.
Task 4: Train machine learning models and normalize features
```

In [178]:
```
X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

In [179]:
```
scaler=MinMaxScaler()
X_train=scaler.fit_transform(X_train)
X_test=scaler.transform(X_test)
```

Artificial Neural Network ¶

In [180]:
```
mlp=MLPClassifier(max_iter=1000)
```

Artificial Neural Network

In [180]:
```
mlp=MLPClassifier(max_iter=1000)
mlp_model=mlp.fit(X_train,y_train)
```

In [181]:
```
mlp_model.score(X_train,y_train)
```

Out[181]: 0.728727104103872

In [182]:
```
mlp_predict=mlp_model.predict(X_test)
```

In [183]:
```
print(classification_report(y_test,mlp_predict))
```

```
              precision    recall  f1-score   support

           f       0.47      0.07      0.13       588
           m       0.74      0.97      0.84      1569

    accuracy                           0.72      2157
   macro avg       0.60      0.52      0.48      2157
weighted avg       0.66      0.72      0.64      2157
```

Decision Tree

In [184]:
```
dt=DecisionTreeClassifier()
dt_model=dt.fit(X_train,y_train)
```

In [185]:
```
dt_model.score(X_train,y_train)
```

Decision Tree

In [184]:
```
dt=DecisionTreeClassifier()
dt_model=dt.fit(X_train,y_train)
```

In [185]:
```
dt_model.score(X_train,y_train)
```

Out[185]: 0.7332483190354742

In [186]:
```
dt_predict=dt_model.predict(X_test)
```

In [187]:
```
print(classification_report(y_test,dt_predict))
```

```
              precision    recall  f1-score   support

           f       0.36      0.05      0.09       588
           m       0.73      0.96      0.83      1569

    accuracy                           0.72      2157
   macro avg       0.54      0.51      0.46      2157
weighted avg       0.63      0.72      0.63      2157
```

Support Vector Classifier

In [188]:
```
svc=SVC()
svc_model=svc.fit(X_train,y_train)
```

In [189]:
```
svc_model.score(X_train,y_train)
```

Out[189]: 0.7245536749362392

In [190]:
```
svc_predict=svc_model.predict(X_test)
```

In [191]:
```
print(classification_report(y_test,svc_predict,zero_division=1))
```

In [191]:
```
print(classification_report(y_test,svc_predict,zero_division=1))
```

```
              precision    recall  f1-score   support

           f       1.00      0.00      0.00       588
           m       0.73      1.00      0.84      1569

    accuracy                           0.73      2157
   macro avg       0.86      0.50      0.42      2157
weighted avg       0.80      0.73      0.61      2157
```

Random Forest

In [192]:
```
rf=RandomForestClassifier()
rf_model=rf.fit(X_train,y_train)
```

In [193]:
```
rf_model.score(X_train,y_train)
```

Out[193]: 0.7332483190354742

In [194]:
```
rf_predict=rf_model.predict(X_test)
```

In [195]:
```
print(classification_report(y_test,rf_predict))
```

```
              precision    recall  f1-score   support

           f       0.38      0.05      0.09       588
           m       0.73      0.97      0.83      1569

    accuracy                           0.72      2157
   macro avg       0.55      0.51      0.46      2157
weighted avg       0.64      0.72      0.63      2157
```

```
Analyze results and draw conclusions.

1.Multilayer Perceptron (MLP):
   - Training Set Accuracy: 72.93%
   - Precision, Recall, F1-scores (per sex): F - 47%, M - 74%
   - The model performs reasonably well, with a higher accuracy and good precision and recall for both sexes.

2.Decision Tree:
   - Training Set Accuracy: 73.32%
   - Precision, Recall, F1-scores (per sex): F - 36%, M - 73%
   - Similar to MLP, the Decision Tree exhibits good accuracy, but precision and recall for females are relatively lower.

3.Support Vector Classifier (SVC):
   - Training Set Accuracy: 72.46%
   - Precision, Recall, F1-scores (per sex): F - 100%, M - 73%
   - Unusual precision of 100% for females might indicate an issue or imbalance in the dataset. Otherwise, the model performs
well.

4.Random Forest:
   - Training Set Accuracy: 73.32%
   - Precision, Recall, F1-scores (per sex): F - 39%, M - 73%
   - Similar to the Decision Tree, Random Forest shows good accuracy, but precision and recall for females are relatively lower.
```

```
Task 5: Compare the performance of different models (ANN, Decision Tree, SVC, Random Forest).

1.Accuracy:
  - MLP: 72.93%
  - Decision Tree: 73.32%
  - SVC: 72.46%
  - Random Forest: 73.32%

2.Precision, Recall, and F1-score:
  - All models demonstrate relatively better performance for predicting males compared to females.
  - SVC shows an unusual precision of 100% for females, which may need further investigation.
  - Overall, Random Forest and Decision Tree perform similarly, and MLP is slightly better in terms of accuracy.

3.Overall Observation:
```

11. Activity 7- Model Comparison and Evaluation

Task 1: Compare the performance of different models.

You trained and evaluated four different machine learning models for predicting income, sex, body type, and zodiac signs:

For Zodiac Sign Prediction:

1. Artificial Neural Network (MLPClassifier):
   - Training Accuracy: 26.76%
   - Test Accuracy: 8.93%
2. Decision Tree:
   - Training Accuracy: 75.76%
   - Test Accuracy: 8.93%
3. Support Vector Classifier (SVC):
   - Training Accuracy: 19.39%
   - Test Accuracy: 8.02%
4. Random Forest:
   - Training Accuracy: 75.76%
   - Test Accuracy: 8.93%

For Body Type Prediction:

For Body Type Prediction:

1. Artificial Neural Network (MLPClassifier):
   - Accuracy: 29%
   - Precision: 0.25, Recall: 0.29, F1-score: 0.37
2. Decision Tree:
   - Accuracy: 26%
   - Precision: 0.23, Recall: 0.26, F1-score: 0.24
3. Support Vector Classifier (SVC):
   - Accuracy: 30%
   - Precision: 0.45, Recall: 0.30, F1-score: 0.23
4. Random Forest:
   - Accuracy: 26%
   - Precision: 0.23, Recall: 0.26, F1-score: 0.24

For Income Prediction:

1. Artificial Neural Network (MLPClassifier):**
   - Best Accuracy: 29.46%
   - Hyperparameters: {'alpha': 0.0001, 'hidden_layer_sizes': (64,), 'random_state': 42}
2. Decision Tree:

- Best Accuracy: 35.36%
- Hyperparameters: {'criterion': 'gini', 'max_depth': 18, 'min_samples_leaf': 8}

3. Support Vector Classifier (SVC):
   - Best Accuracy: 36.31%
   - Hyperparameters: {'kernel': 'linear', 'random_state': 42}
4. Random Forest:
   - Best Accuracy: 36.30%
   - Hyperparameters: {'criterion': 'entropy', 'max_depth': 24, 'min_samples_leaf': 7, 'n_estimators': 100}

For Sex Prediction:

1. Artificial Neural Network (MLPClassifier):
   - Accuracy: 72.93%
2. Decision Tree:
   - Accuracy: 72.33%
3. Support Vector Classifier (SVC):
   - Accuracy: 72.45%
4. Random Forest:
   - Accuracy: 72.33%

Task 2: Evaluate accuracy, precision, recall, and F1-score of each model.

For Zodiac Sign Prediction:

- All models exhibited low accuracy, precision, recall, and F1-scores for each sign, suggesting challenges in capturing underlying patterns.

For Body Type Prediction:

- Decision Tree and Random Forest exhibited slightly better overall performance compared to MLP and SVC.
- Precision, recall, and F1-scores fluctuated, emphasizing the need for further model refinement.

For Income Prediction:

- Decision Tree and Random Forest outperformed the Artificial Neural Network and Support Vector Classifier.
- Precision, recall, and F1-score varied across income categories, indicating challenges in predicting specific income levels.

For Sex Prediction:

- All models achieved similar accuracies around 72%, with little variation.
- Precision, recall, and F1-score were reasonably balanced between the classes.

Task 3: Identify strengths and weaknesses of each model.

Strengths:

- Decision Tree and Random Forest showed competitive performance for income, sex, and body type prediction.
- Neural networks (MLP) captured complex patterns for sex and body type prediction.

Weaknesses:

- Predicting income levels proved challenging, possibly due to the complexity of the underlying patterns.
- Support Vector Classifier struggled with precision in sex prediction, body type prediction, and zodiac sign prediction.

Task 4: Summarize model comparison results.

- Decision Tree and Random Forest are recommended for income, sex, and body type prediction, considering their balanced performance.
- Neural networks performed well in sex and body type prediction but struggled with income prediction.
- Support Vector Classifier had lower precision in sex prediction, body type prediction, and zodiac sign prediction.

Task 5: Provide recommendations based on model performance.

1.For Zodiac Sign Prediction:

- The overall performance of all models for zodiac sign prediction is low.
- Further analysis, feature engineering, or hyperparameter tuning may be required to improve model performance.

Task 5: Provide recommendations based on model performance.

1.For Zodiac Sign Prediction:

- The overall performance of all models for zodiac sign prediction is low.
- Further analysis, feature engineering, or hyperparameter tuning may be required to improve model performance.

2.For Body Type Prediction:

- Decision Tree and Random Forest exhibited better performance, but further refinement is needed.
- Explore additional features or conduct hyperparameter tuning to improve model accuracy.

3..For Income Prediction:

- Consider Decision Tree or Random Forest for better performance.
- Further feature engineering or exploration may improve model performance.

4.For Sex Prediction:

- Any of the models could be suitable, but Decision Tree and Random Forest are slightly favored.
- Evaluate if additional features could enhance prediction accuracy.

5.General Recommendations:

- Continue refining models and exploring new features to improve predictions.
- Regularly monitor and update models as new data becomes available.

12. Conclusions and Future Improvements

Conclusions:

Based on our analysis of the OkCupid dataset, we found that while Decision Tree and Random Forest models performed better in identifying sex and wealth categories, they had trouble detecting body types and zodiac signs. The complexity of the changes in precision, recall, and F1 scores in these categories highlights the significance of continued work in feature engineering. Sex prediction models, particularly those based on Random Forests and Decision Trees, showed balanced performance in precision, recall, and F1-score, showing reliability in their predictions, despite reaching identical accuracies of roughly 72%. However, variations in these measures point to the need for more feature considerations or model improvement for predictions related to body type and zodiac signs.

Future Enhancements:

The following areas should be the focus of future advancements to significantly increase predicted accuracy. A more thorough grasp of the factors influencing income can be obtained for income prediction by investigating additional socioeconomic characteristics and incorporating data from outside sources. Performance improvements could come from experimenting with gradient-boosting algorithms or sophisticated ensemble techniques. Examining certain behavioral or demographic characteristics may improve model performance in the context of sex prediction, and taking into account ensemble techniques or deep learning architectures may result in increased accuracy. Detailed feature analysis is advised for body type prediction to find underrepresented or missing elements that affect predictions. The model could be improved by experimenting with more complex neural network topologies or ensemble methods. Lastly, reassessing features for zodiac sign prediction.

In conclusion, reassessing features and investigating different ways to encode astrological information, as well as adding outside data on astrological trends or user traits, could enhance the predictive power of zodiac sign prediction. Comprehensive hyperparameter tweaking, investigation of sophisticated feature engineering methods, and routine model updates and retraining in response to fresh data are examples of general model advances. A more thorough understanding of model performance, user satisfaction, and ethical considerations will result from the use of various assessment metrics, cross-validation procedures, user feedback collection, and iterative development processes. The prediction models' long-term success will depend on their constant observation and adjustment to changing data patterns.