In [74]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [75]:

```
df=pd.read_csv(r"C:\Users\balakumar\OneDrive\Desktop\dataset\work.csv")
```

In [76]:

```
df
```

Out[76]:

	Res. No	Age	Gender	Marital status	Education	Income	Covid- infected or not	CAS1	CAS2	CAS3	CAS4	CAS5
0	1	2	1	1	1	1	1	1	1	2	2	2
1	2	1	0	1	1	2	1	1	1	2	2	2
2	3	2	0	1	3	1	1	3	3	0	0	3
3	4	3	0	0	1	2	1	0	1	0	1	0
4	5	2	0	1	1	1	1	1	2	1	1	2
...	...	...	...	...	...	...	...	...	...	...	...	...
1345	1346	2	0	0	2	2	1	3	2	1	3	3
1346	1347	3	0	0	1	2	1	3	3	3	3	3
1347	1348	3	0	0	1	2	0	1	0	1	1	2
1348	1349	1	1	0	1	2	1	3	2	1	2	3
1349	1350	2	0	1	1	1	0	2	0	0	0	1

1350 rows × 12 columns

In [77]:

```
df.shape
```

Out[77]:

```
(1350, 12)
```

In [78]:
```
df.isnull().sum()
```

Out[78]:
```
Res. No                 0
Age                     0
Gender                  0
Marital status          0
Education               0
Income                  0
Covid- infected or not  0
CAS1                    0
CAS2                    0
CAS3                    0
CAS4                    0
CAS5                    0
dtype: int64
```

In [79]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1350 entries, 0 to 1349
Data columns (total 12 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   Res. No                 1350 non-null   int64 
 1   Age                     1350 non-null   int64 
 2   Gender                  1350 non-null   int64 
 3   Marital status          1350 non-null   int64 
 4   Education               1350 non-null   int64 
 5   Income                  1350 non-null   int64 
 6   Covid- infected or not  1350 non-null   int64 
 7   CAS1                    1350 non-null   object
 8   CAS2                    1350 non-null   int64 
 9   CAS3                    1350 non-null   int64 
 10  CAS4                    1350 non-null   int64 
 11  CAS5                    1350 non-null   int64 
dtypes: int64(11), object(1)
memory usage: 126.7+ KB
```

In [80]:
```
df.columns
```

Out[80]:
```
Index(['Res. No', 'Age', 'Gender', 'Marital status', 'Education', 'Income',
       'Covid- infected or not', 'CAS1', 'CAS2', 'CAS3', 'CAS4', 'CAS5'],
      dtype='object')
```

In [81]:
```
#To check correlation between vectors
correlation = df.corr()
```

In [82]:
```
correlation
```

Out[82]:

	Res. No	Age	Gender	Marital status	Education	Income	Covid- infected or not	CAS2	CAS3	CAS4	CAS5
Res. No	1.000000	0.059598	-0.030123	-0.067225	-0.002179	0.055200	-0.007846	0.015461	0.046159	0.042492	0.014578
Age	0.059598	1.000000	0.044289	-0.080058	-0.058806	0.084014	-0.030567	0.009756	0.042519	0.029378	0.020434
Gender	-0.030123	0.044289	1.000000	0.032329	-0.093072	-0.019621	-0.000768	0.035036	0.022752	0.027767	-0.000631
Marital status	-0.067225	-0.080058	0.032329	1.000000	0.008777	-0.661565	0.043465	0.014007	0.029137	0.026799	-0.054360
Education	-0.002179	-0.058806	-0.093072	0.008777	1.000000	-0.006456	0.021575	-0.002785	-0.013849	-0.012464	0.016278
Income	0.055200	0.084014	-0.019621	-0.661565	-0.006456	1.000000	-0.003209	-0.053953	0.003141	-0.035159	-0.013909
Covid- infected or not	-0.007846	-0.030567	-0.000768	0.043465	0.021575	-0.003209	1.000000	0.020668	-0.012125	-0.022504	-0.044196
CAS2	0.015461	0.009756	0.035036	0.014007	-0.002785	-0.053953	0.020668	1.000000	0.319270	0.370506	0.137341
CAS3	0.046159	0.042519	0.022752	0.029137	-0.013849	0.003141	-0.012125	0.319270	1.000000	0.263840	0.152906
CAS4	0.042492	0.029378	0.027767	0.026799	-0.012464	-0.035159	-0.022504	0.370506	0.263840	1.000000	0.118909
CAS5	0.014578	0.020434	-0.000631	-0.054360	0.016278	-0.013909	-0.044196	0.137341	0.152906	0.118909	1.000000

In [83]:

```
gender = pd.crosstab(index = df["Gender"],
                    columns = 'count',
                    normalize = True)
print(gender)
```

```
col_0      count
Gender
0       0.561481
1       0.432593
4       0.005185
5       0.000741
```

In [84]:

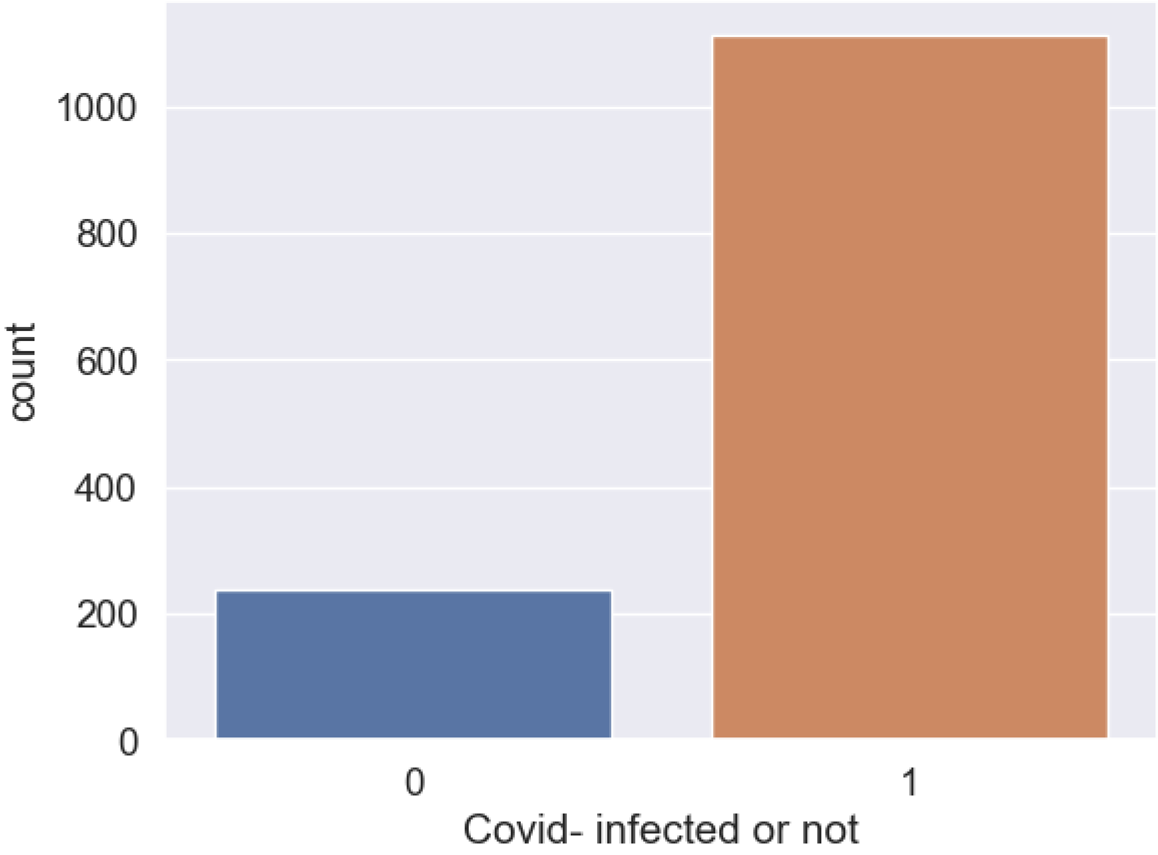
```
#Covid-19 infected migrant workers with gender proportion
gender_stat = pd.crosstab(index = df['Gender'],columns =df['Covid- infected or not'],margins = True, normalize = 'inde
print(gender_stat)
```

```
Covid- infected or not         0         1
Gender
0                       0.174142  0.825858
1                       0.179795  0.820205
4                       0.142857  0.857143
5                       0.000000  1.000000
All                     0.176296  0.823704
```

```
In [85]: covid_19_infected = sns.countplot(df['Covid- infected or not'])
         covid_19_infected
```

```
C:\Users\balakumar\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data
`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
```

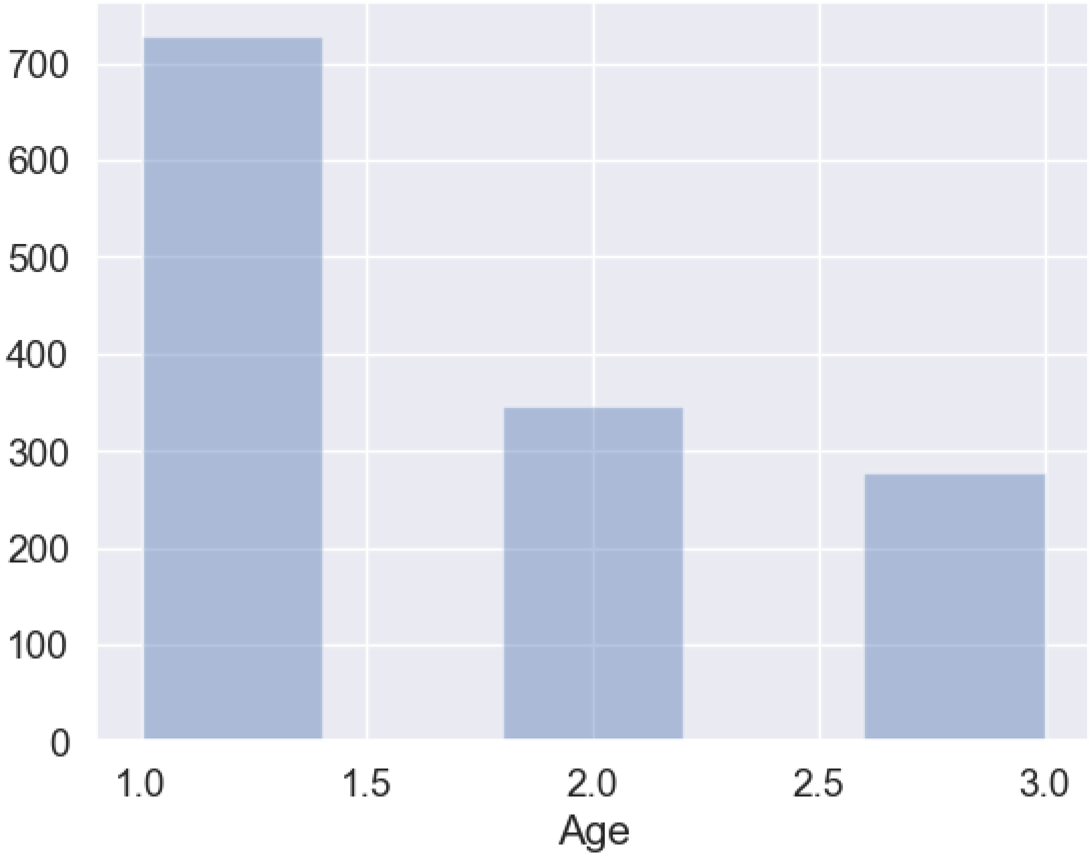
```
Out[85]: <AxesSubplot:xlabel='Covid- infected or not', ylabel='count'>
```



In [86]:
```
#
sns.distplot(df['Age'],bins = 5, kde = False)
```

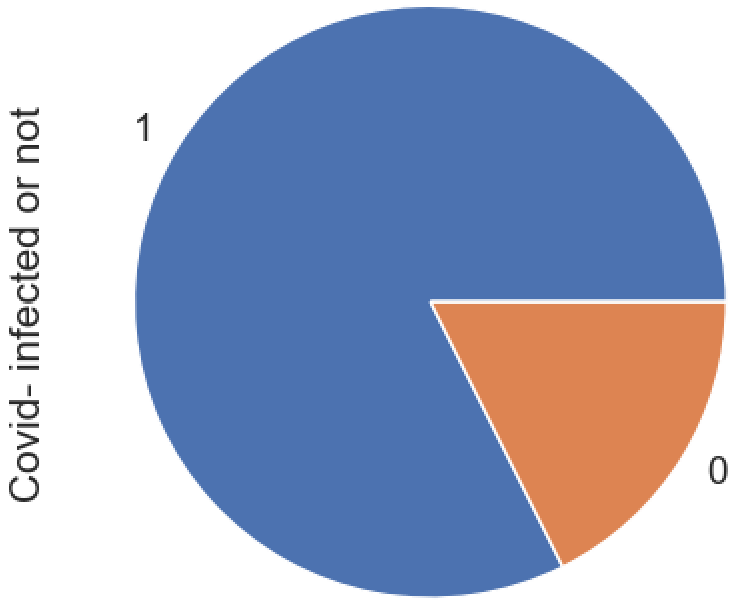
```
C:\Users\balakumar\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `di
splot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
```

Out[86]:
```
<AxesSubplot:xlabel='Age'>
```



```
In [87]: df["Covid- infected or not"].value_counts().plot(kind="pie")
```

```
Out[87]: <AxesSubplot:ylabel='Covid- infected or not'>
```

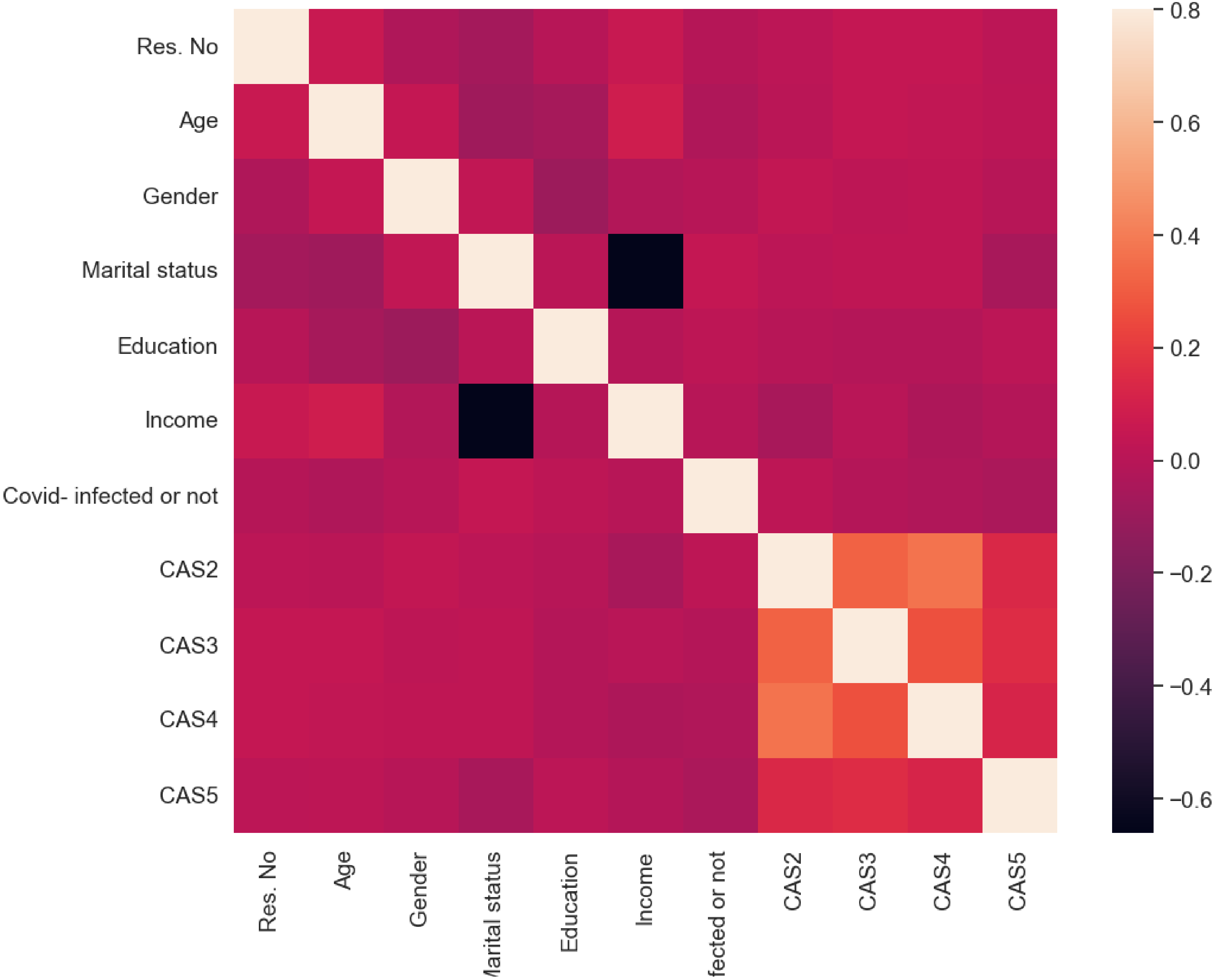


In [88]:

```
cormat = df.corr()
f, ax = plt.subplots(figsize=(12,9))
sns.heatmap(cormat, vmax=.8, square=True);
plt.show
```

Out[88]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```


Res. No
Age
Gender
Marital status
Education
Income
Covid- infected or not
CAS2
CAS3
CAS4
CAS5
Res. No
Age
Gender
/arital status
Education
Income
fected or not
CAS2
CAS3
CAS4
CAS5
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

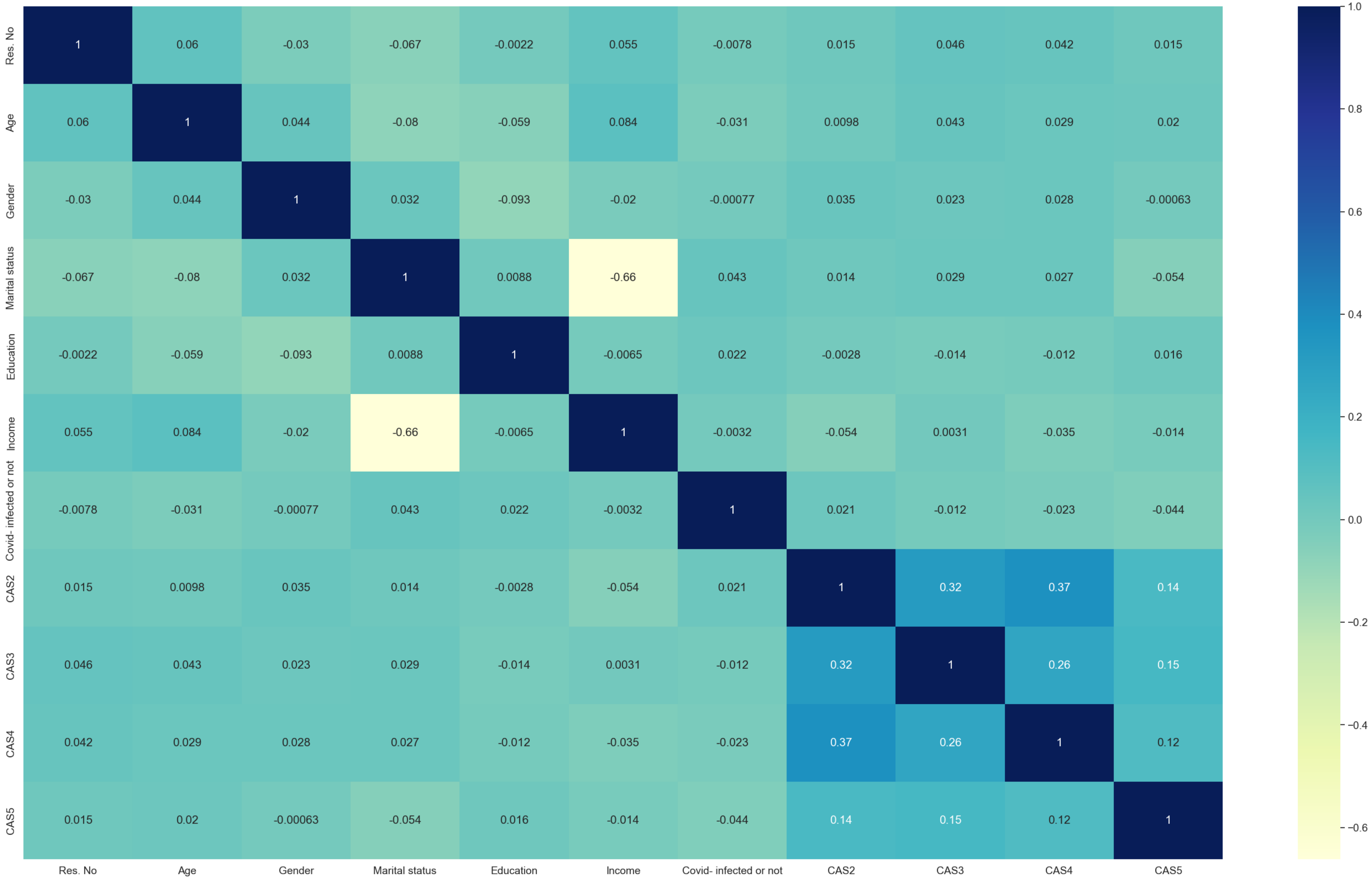
N
Covid- in

In [89]:

```
plt.figure(figsize=(35,20))
sns.heatmap(df.corr(), annot=True, cmap="YlGnBu")
```

Out[89]:

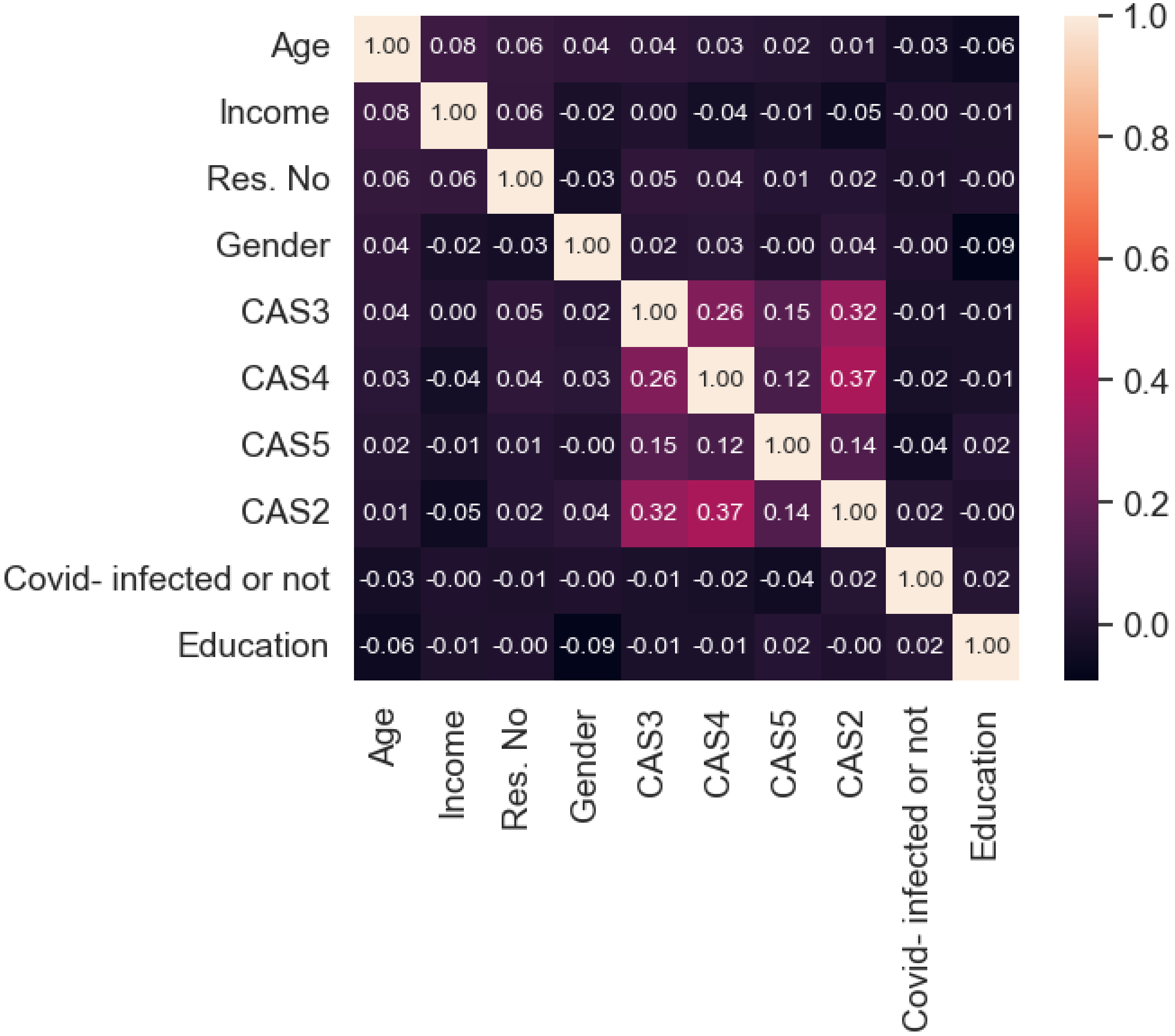
```
<AxesSubplot:>
```



	Res. No	Age	Gender	Marital status	Education	Income	Covid- infected or not	CAS2	CAS3	CAS4	CAS5
Res. No	1	0.06	-0.03	-0.067	-0.0022	0.055	-0.0078	0.015	0.046	0.042	0.015
Age	0.06	1	0.044	-0.08	-0.059	0.084	-0.031	0.0098	0.043	0.029	0.02
Gender	-0.03	0.044	1	0.032	-0.093	-0.02	-0.00077	0.035	0.023	0.028	-0.00063
Marital status	-0.067	-0.08	0.032	1	0.0088	-0.66	0.043	0.014	0.029	0.027	-0.054
Education	-0.0022	-0.059	-0.093	0.0088	1	-0.0065	0.022	-0.0028	-0.014	-0.012	0.016
Income	0.055	0.084	-0.02	-0.66	-0.0065	1	-0.0032	-0.054	0.0031	-0.035	-0.014
Covid- infected or not	-0.0078	-0.031	-0.00077	0.043	0.022	-0.0032	1	0.021	-0.012	-0.023	-0.044
CAS2	0.015	0.0098	0.035	0.014	-0.0028	-0.054	0.021	1	0.32	0.37	0.14
CAS3	0.046	0.043	0.023	0.029	-0.014	0.0031	-0.012	0.32	1	0.26	0.15
CAS4	0.042	0.029	0.028	0.027	-0.012	-0.035	-0.023	0.37	0.26	1	0.12
CAS5	0.015	0.02	-0.00063	-0.054	0.016	-0.014	-0.044	0.14	0.15	0.12	1

In [90]:

```
#treatment correlation matrix
k = 10
cols = cormat.nlargest(k, 'Age')['Age'].index
cm = np.corrcoef(df[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 10},yticklabels=cols, xticklabe
plt.show()
```



	Age	Income	Res. No	Gender	CAS3	CAS4	CAS5	CAS2	Covid- infected or not	Education
Age	1.00	0.08	0.06	0.04	0.04	0.03	0.02	0.01	-0.03	-0.06
Income	0.08	1.00	0.06	-0.02	0.00	-0.04	-0.01	-0.05	-0.00	-0.01
Res. No	0.06	0.06	1.00	-0.03	0.05	0.04	0.01	0.02	-0.01	-0.00
Gender	0.04	-0.02	-0.03	1.00	0.02	0.03	-0.00	0.04	-0.00	-0.09
CAS3	0.04	0.00	0.05	0.02	1.00	0.26	0.15	0.32	-0.01	-0.01
CAS4	0.03	-0.04	0.04	0.03	0.26	1.00	0.12	0.37	-0.02	-0.01
CAS5	0.02	-0.01	0.01	-0.00	0.15	0.12	1.00	0.14	-0.04	0.02
CAS2	0.01	-0.05	0.02	0.04	0.32	0.37	0.14	1.00	0.02	-0.00
Covid- infected or not	-0.03	-0.00	-0.01	-0.00	-0.01	-0.02	-0.04	0.02	1.00	0.02
Education	-0.06	-0.01	-0.00	-0.09	-0.01	-0.01	0.02	-0.00	0.02	1.00

Some charts to see data relationship

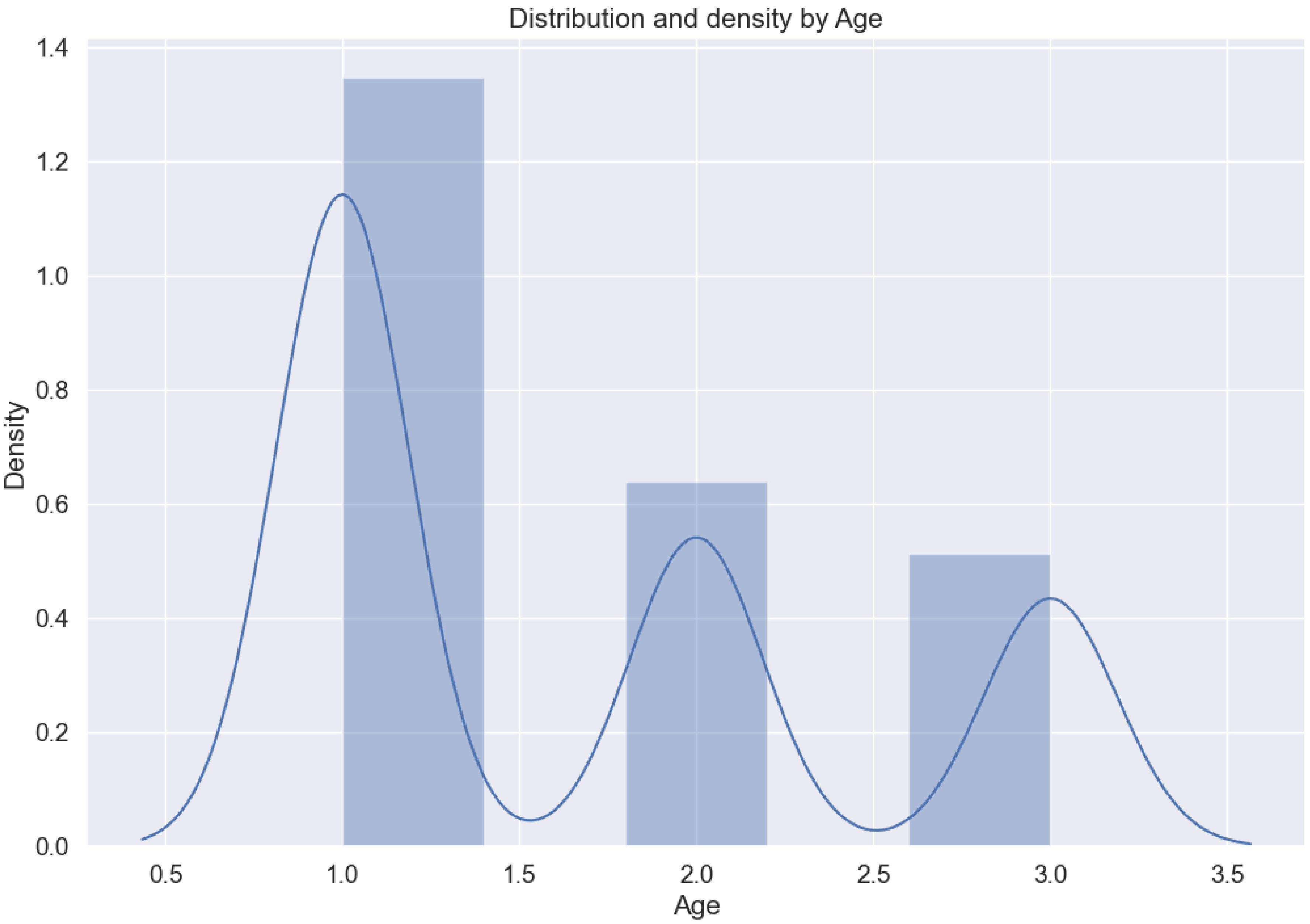
In [91]:

```
#Distribution and density by Age
plt.figure(figsize=(12,8))
sns.distplot(df['Age'],bins=5)
plt.title("Distribution and density by Age")
plt.xlabel('Age')
```

```
C:\Users\balakumar\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `di
splot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
```

Out[91]:

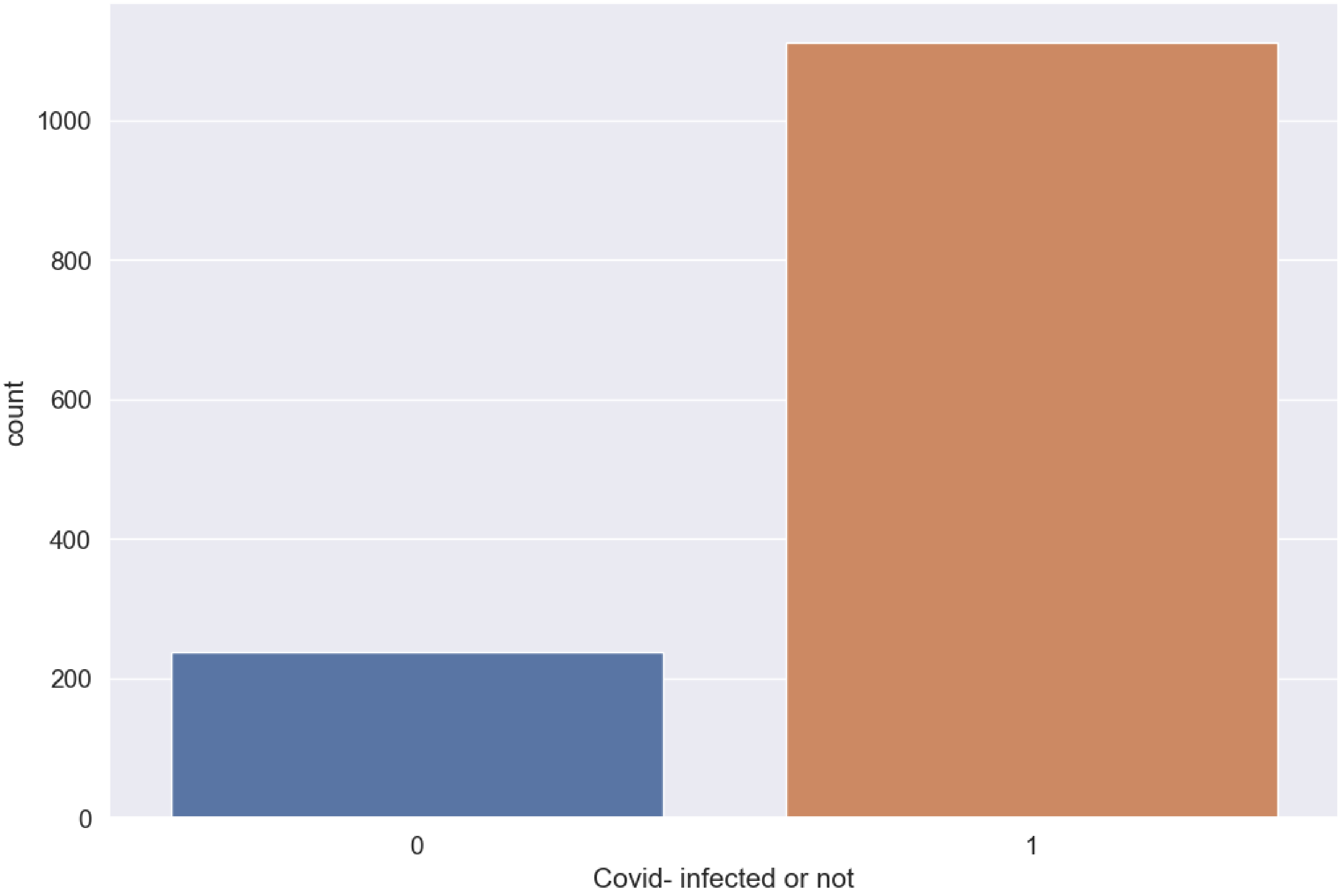
```
Text(0.5, 0, 'Age')
```


Distribution and density by Age
Density
Age
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0.5
1.0
1.5
2.0
2.5
3.0
3.5

In [92]:

```
plt.figure(figsize=(12,8))
labels = df['Age']
g = sns.countplot(x='Covid- infected or not', data = df)
g.set_xticklabels(labels)
plt.title("Total Distribution by Covid impacted or not")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_8180\4257121072.py in <module>
      2 labels = df['Age']
      3 g = sns.countplot(x='Covid- infected or not', data = df)
----> 4 g.set_xticklabels(labels)
      5 plt.title("Total Distribution by Covid impacted or not")

~\anaconda3\lib\site-packages\matplotlib\axes\_base.py in wrapper(self, *args, **kwargs)
     73 
     74         def wrapper(self, *args, **kwargs):
---> 75             return get_method(self)(*args, **kwargs)
     76 
     77         wrapper.__module__ = owner.__module__

~\anaconda3\lib\site-packages\matplotlib\axis.py in _set_ticklabels(self, labels, fontdict, minor, **kwargs)
   1796         if fontdict is not None:
   1797             kwargs.update(fontdict)
-> 1798         return self.set_ticklabels(labels, minor=minor, **kwargs)
   1799 
   1800     def _set_tick_locations(self, ticks, *, minor=False):

~\anaconda3\lib\site-packages\matplotlib\axis.py in set_ticklabels(self, ticklabels, minor, **kwargs)
   1718             # remove all tick labels, so only error for > 0 ticklabels
   1719             if len(locator.locs) != len(ticklabels) and len(ticklabels) != 0:
-> 1720                 raise ValueError(
   1721                     "The number of FixedLocator locations"
   1722                     f" ({len(locator.locs)}), usually from a call to"

ValueError: The number of FixedLocator locations (2), usually from a call to set_ticks, does not match the number of
ticklabels (1350).
```

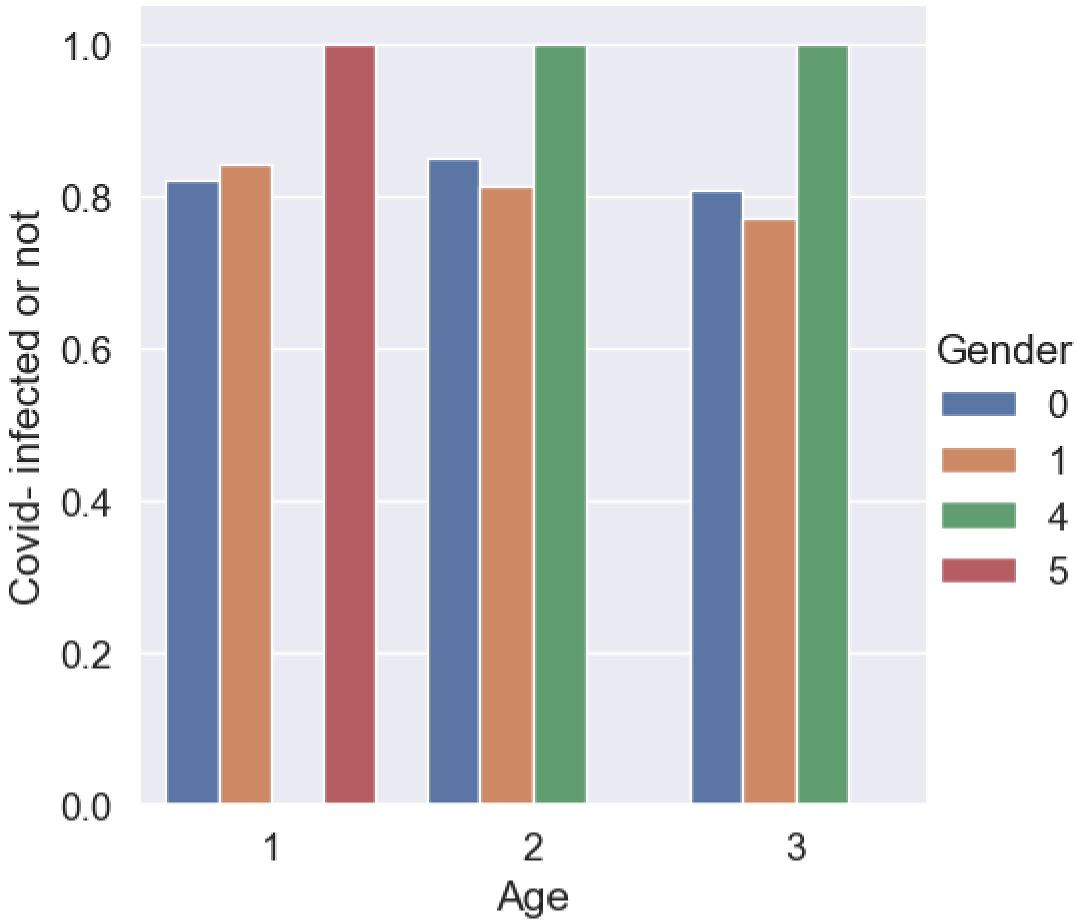

1000
800
600
400
200
0
count
0
1
Covid- infected or not

In [93]:

```
o = df["Age"]
g = sns.factorplot(x='Age', y='Covid- infected or not', hue='Gender', data = df, kind="bar", ci=None)
g.set_xticklabels(o)

plt.title('Probability of mental health condition')
plt.ylabel('Probability x 100')
plt.xlabel('Age')


#replace legend labels
new_labels = df["Gender"]
for t,l in zip(g._legend.texts, new_labels):t.set_text(l)

#positioning the legend
g.fig.subplots_adjust(top=0.9, right=0.9)
plt.show()
```

```
C:\Users\balakumar\anaconda3\lib\site-packages\seaborn\categorical.py:3717: UserWarning:

The `factorplot` function has been renamed to `catplot`. The original name will be removed in a future release. Pleas
e update your code. Note that the default `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_8180\791786080.py in <module>
      1 o = df["Age"]
      2 g = sns.factorplot(x='Age', y='Covid- infected or not', hue='Gender', data = df, kind="bar", ci=None)
----> 3 g.set_xticklabels(o)
      4 
      5 plt.title('Probability of mental health condition')

~\anaconda3\lib\site-packages\seaborn\axisgrid.py in set_xticklabels(self, labels, step, **kwargs)
    880                 ax.set_xticklabels(curr_labels, **kwargs)
    881             else:
--> 882                 ax.set_xticklabels(labels, **kwargs)
    883         return self
    884 

~\anaconda3\lib\site-packages\matplotlib\axes\_base.py in wrapper(self, *args, **kwargs)
     73 
     74         def wrapper(self, *args, **kwargs):
---> 75             return get_method(self)(*args, **kwargs)
     76 
     77         wrapper.__module__ = owner.__module__

~\anaconda3\lib\site-packages\matplotlib\axis.py in _set_ticklabels(self, labels, fontdict, minor, **kwargs)
   1796         if fontdict is not None:
   1797             kwargs.update(fontdict)
-> 1798         return self.set_ticklabels(labels, minor=minor, **kwargs)
   1799 
   1800     def _set_tick_locations(self, ticks, *, minor=False):

~\anaconda3\lib\site-packages\matplotlib\axis.py in set_ticklabels(self, ticklabels, minor, **kwargs)
   1718             # remove all tick labels, so only error for > 0 ticklabels
   1719             if len(locator.locs) != len(ticklabels) and len(ticklabels) != 0:
-> 1720                 raise ValueError(
   1721                     "The number of FixedLocator locations"
   1722                     f" ({len(locator.locs)}), usually from a call to"

ValueError: The number of FixedLocator locations (3), usually from a call to set_ticks, does not match the number of
ticklabels (1350).
```



```
In [94]: df.rename(columns={'Covid- infected or not':'Covid- infected'}, inplace=True)
```

In [95]:

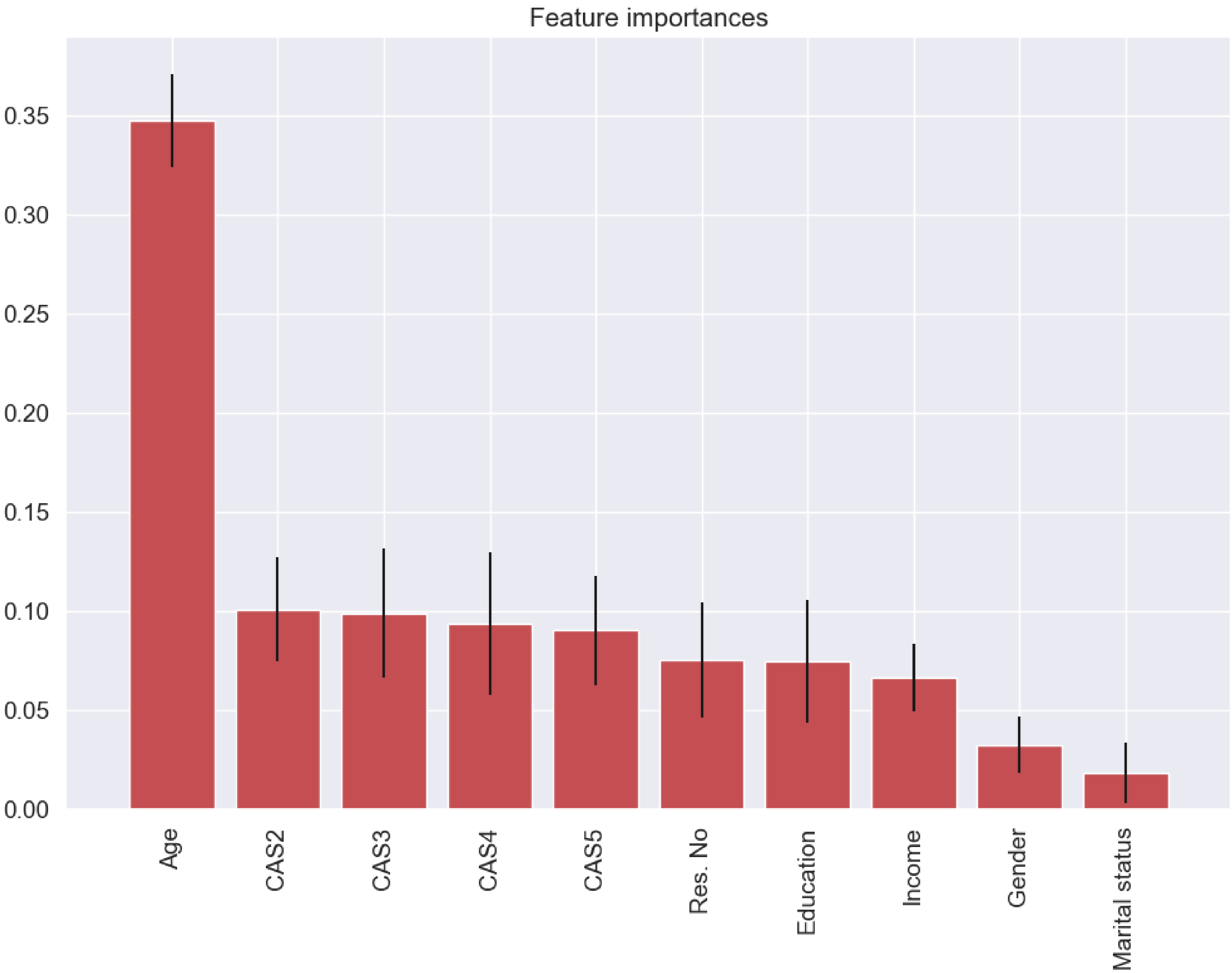
```
#To Visualize data
import seaborn as sns
#To partition the data
from sklearn.model_selection import train_test_split
#Importing library for logistic regression
from sklearn.linear_model import LogisticRegression
#Importing performance metrics -accuracy score & confussion matrix
from sklearn.metrics import accuracy_score,confusion_matrix

from sklearn.ensemble import ExtraTreesClassifier
```

In [96]:

```
feature_cols = ['Age','CAS2','CAS3','CAS4','CAS5','Res. No','Education','Income','Gender','Marital status']
X = df[feature_cols]
y = df['Covid- infected'].astype('float')

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3, random_state=0)

#create a dict for final graph
#Use : methodDict['Stacking'] = accuracy_score
methodDict = {}
rmseDict = ()
```

In [97]:

```
forest = ExtraTreesClassifier(n_estimators=250, random_state=0)

forest.fit(X,y)
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_],
            axis=0)
indices = np.argsort(importances)[::-1]

labels = []
for f in range(X.shape[1]):
    labels.append(feature_cols[f])

plt.figure(figsize=(12,8))
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices],
       color = 'r', yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), labels, rotation='vertical')
plt.xlim([-1,X.shape[1]])
plt.show()
```


Feature importances
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Age
CAS2
CAS3
CAS4
CAS5
Res. No
Education
Income
Gender
Marital status

Evaluating models

In [98]:

```
"""Logistic Regression"""
```

Out[98]:

```
'Logistic Regression'
```

In [99]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, mean_squared_error, precision_recall_curve
from sklearn.model_selection import cross_val_score
from sklearn import metrics

from scipy.stats import randint

#Importing performance metrics -accuracy score & confussion matrix
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [100]:

```
data = df.copy()
```

In [101]:

```
data.shape
```

Out[101]:

```
(1350, 12)
```

In [102]:
```
data["Covid- infected"] = pd.get_dummies(data["Covid- infected"])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_8180\934390265.py in <module>
----> 1 data["Covid- infected"] = pd.get_dummies(data["Covid- infected"])

~\anaconda3\lib\site-packages\pandas\core\frame.py in __setitem__(self, key, value)
   3643             self._setitem_array(key, value)
   3644         elif isinstance(value, DataFrame):
-> 3645             self._set_item_frame_value(key, value)
   3646         elif (
   3647             is_list_like(value)

~\anaconda3\lib\site-packages\pandas\core\frame.py in _set_item_frame_value(self, key, value)
   3773             len_cols = 1 if is_scalar(cols) else len(cols)
   3774             if len_cols != len(value.columns):
-> 3775                 raise ValueError("Columns must be same length as key")
   3776 
   3777             # align right-hand-side columns if self.columns

ValueError: Columns must be same length as key
```

In [103]:
```
data.shape
```

Out[103]:
```
(1350, 12)
```

In [104]:
```
new_data = pd.get_dummies(data , drop_first = True)
new_data.head()
```

Out[104]:

	Res. No	Age	Gender	Marital status	Education	Income	Covid- infected	CAS2	CAS3	CAS4	CAS5	CAS1_1	CAS1_2	CAS1_3	CAS1_c
0	1	2	1	1	1	1	1	1	2	2	2	1	0	0	0
1	2	1	0	1	1	2	1	1	2	2	2	1	0	0	0
2	3	2	0	1	3	1	1	3	0	0	3	0	0	1	0
3	4	3	0	0	1	2	1	1	0	1	0	0	0	0	0
4	5	2	0	1	1	1	1	2	1	1	2	1	0	0	0

In [105]:
```
columns_list = list(new_data.columns)
print(columns_list)
```

```
['Res. No', 'Age', 'Gender', 'Marital status', 'Education', 'Income', 'Covid- infected', 'CAS2', 'CAS3', 'CAS4', 'CAS
5', 'CAS1_1', 'CAS1_2', 'CAS1_3', 'CAS1_c']
```

In [106]:
```
features = list(set(columns_list)-set(['Covid- infected']))
print(features)
```

```
['Income', 'CAS1_3', 'CAS5', 'CAS1_1', 'CAS1_2', 'CAS3', 'CAS1_c', 'Res. No', 'Age', 'Gender', 'CAS2', 'CAS4', 'Educa
tion', 'Marital status']
```

In [107]:
```
y = new_data['Covid- infected'].values
print(y)
```

```
[1 1 1 ... 0 1 0]
```

In [108]:

```
x = new_data[features].values
print(x)
```

```
[[1 0 2 ... 2 1 1]
 [2 0 2 ... 2 1 1]
 [1 1 3 ... 0 3 1]
 ...
 [2 0 2 ... 1 1 0]
 [2 1 3 ... 2 1 0]
 [1 0 1 ... 0 1 1]]
```

In [109]:

```
x.shape
```

Out[109]:

```
(1350, 14)
```

In [110]:

```
y.shape
```

Out[110]:

```
(1350,)
```

In [111]:

```
train_x,test_x,train_y,test_y = train_test_split(x,y,test_size=0.3, random_state=0)
```

In [112]:

```
train_x.shape
```

Out[112]:

```
(945, 14)
```

In [113]:

```
train_y.shape
```

Out[113]:

```
(945,)
```

In [114]:

```
test_x.shape
```

Out[114]:

```
(405, 14)
```

In [115]:
```
test_y.shape
```

Out[115]:
```
(405,)
```

In [116]:
```
#make instance of the model
logistic = LogisticRegression()
```

In [117]:
```
logistic.fit(train_x,train_y)
logistic.coef_
```

```
C:\Users\balakumar\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning:

lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessin
g.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/mo
dules/linear_model.html#logistic-regression)
```

Out[117]:
```
array([[ 4.22222560e-01,  1.12845534e-01, -1.00519405e-01,
         1.74817364e-01, -1.19710481e-01,  2.91020158e-03,
         1.42915016e-02,  8.83981370e-05, -1.06753598e-02,
        -4.02282521e-03,  5.81813176e-02, -7.39898001e-02,
         1.45447564e-01,  7.52157425e-01]])
```

In [45]:
```
logistic.intercept_
```

Out[45]:
```
array([0.41789231])
```

In [46]:
```
prediction = logistic.predict(test_x)
print(prediction)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

In [47]:
```
confusion_matrix = confusion_matrix(test_y, prediction)
print(confusion_matrix)
```

```
[[  0  78]
 [  0 327]]
```

In [48]:
```
# Calculating the accuracy

accuracy_score = accuracy_score(test_y,prediction)
accuracy_score
```

Out[48]:
```
0.8074074074074075
```

RANDOM FOREST

In [49]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
     x, y, test_size=0.2, random_state=42)
```

In [50]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [51]:

```
classifierclf = RandomForestClassifier()
```

In [52]:

```
classifierclf.fit(x_train,y_train)
```

Out[52]:

```
RandomForestClassifier()
```

In [53]:

```
x_train.shape
```

Out[53]:

```
(1080, 14)
```

In [54]:

```
y_pred = classifierclf.predict(x_test)
```

In [55]:

```
#perform standziation
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[55]:

```
0.8296296296296296
```

PCA

In [56]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [57]:

```
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

In [58]:

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 10)
```

In [59]:
```
x_train_trf = pca.fit_transform(x_train)
x_test_trf = pca.transform(x_test)
```

In [60]:
```
x_train_trf.shape
```

Out[60]:
```
(1080, 10)
```

In [61]:
```
x_train_trf.shape
```

Out[61]:
```
(1080, 10)
```

In [62]:
```
classifierclf = RandomForestClassifier()
```

In [63]:
```
classifierclf.fit(x_train_trf,y_train)
```

Out[63]:
```
RandomForestClassifier()
```

In [64]:
```
y_pred = classifierclf.predict(x_test_trf)
```

In [65]:
```
accuracy_score(y_test,y_pred)
```

Out[65]:
```
0.8185185185185185
```

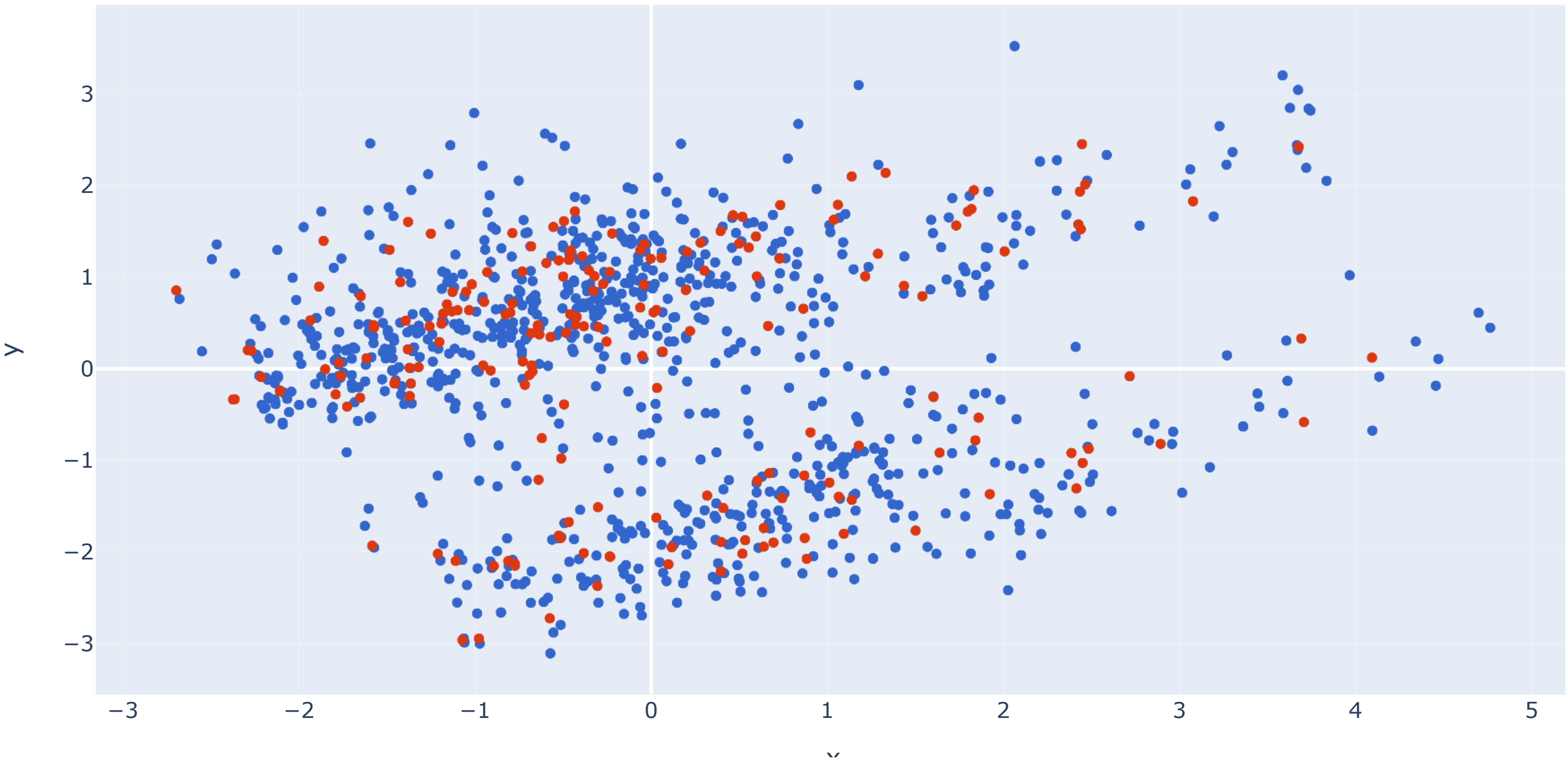
In [66]:
```
pca = PCA(n_components=3)
x_train_trf = pca.fit_transform(x_train)
x_test_trf = pca.transform(x_test)
```

```
In [67]: x_train_trf
```

```
Out[67]: array([[-0.800831  ,  0.46627249,  1.81862815],
                [-1.45561388, -0.14720139,  0.83843941],
                [-0.06217479,  0.66876324,  1.84793879],
                ...,
                [ 4.34139027,  0.2973388 ,  0.25442508],
                [-0.4374745 ,  0.94716784,  0.58549119],
                [-2.27695521,  0.27376842,  0.75364073]])
```

In [68]:

```
import plotly.express as px
y_train_trf = y_train.astype(str)
fig = px.scatter(x=x_train_trf[:,0],
                y=x_train_trf[:,1],
                color = y_train_trf,
                color_discrete_sequence = px.colors.qualitative.G10
                )
fig.show()
```



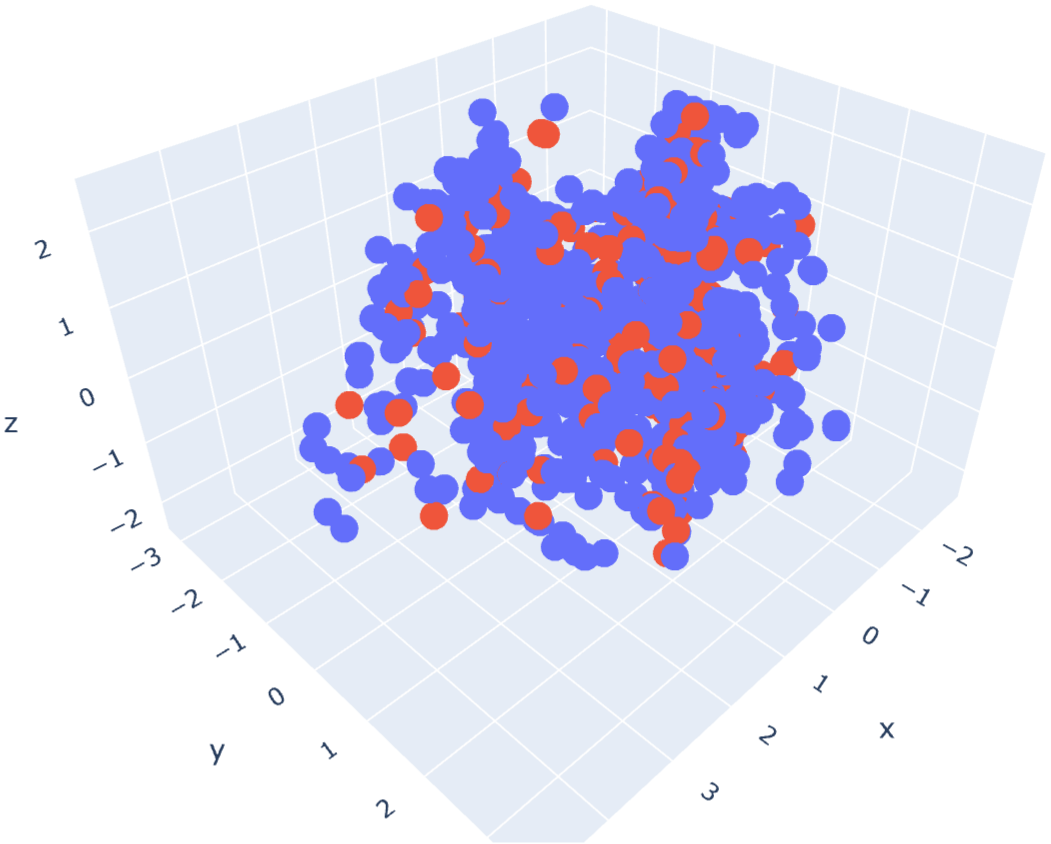
In [69]:

```
pca = PCA(n_components=3)
x_train_trf = pca.fit_transform(x_train)
x_test_trf = pca.transform(x_test)
```

```
In [70]: x_train_trf
```

```
Out[70]: array([[-0.80036973,  0.46743209,  1.82025925],
                [-1.45580267, -0.14723885,  0.83708358],
                [-0.06187977,  0.66949332,  1.84898979],
                ...,
                [ 4.34148159,  0.29751105,  0.25482023],
                [-0.43788173,  0.94660988,  0.58329507],
                [-2.2771019 ,  0.27388197,  0.75239641]])
```

In [71]:
```
y_train_trf = y_train.astype(str)
fig = px.scatter_3d(data, x=x_train_trf[:,0],
                y=x_train_trf[:,1],
                z=x_train_trf[:,2],
                color = y_train_trf)

fig.update_layout(margin=dict(l=20, r=20, t=20, b=20))
fig.show()
```



In [72]:

```
#Eigen values
pca.explained_variance_
```

Out[72]:

```
array([1.87761549, 1.6960011 , 1.37328951])
```

In [73]:

```
#finding optimum number of principle components
#to check percentage of variance explained by individual eigen vector

pca.explained_variance_ratio_

#eigen vector 1:13%
#eigen vector 2:12%
#eigen vector 3:09%..................
```

Out[73]:

```
array([0.13399121, 0.12103077, 0.09800128])
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: