In [74]:

```
import pandas as pd
import sklearn.model_selection
import sklearn.tree
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score,precision_score, recall_score, accuracy_score,

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import GridSearchCV, learning_curve, KFold, cross_val_
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder, StandardScaler
import random
from sklearn.svm import SVC
import sklearn.metrics as sk
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.model_selection import ShuffleSplit
from sklearn.svm import SVC
from sklearn.model_selection import learning_curve
```

In [49]:

```
data = pd.read_csv('bank_data.csv', delimiter=';')
```

In [50]:

```
data.head()
```

Out[50]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon
1	57	services	married	high.school	unknown	no	no	telephone	may	mon
2	37	services	married	high.school	no	yes	no	telephone	may	mon
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon
4	56	services	married	high.school	no	no	yes	telephone	may	mon

5 rows × 21 columns

In [51]:

```
data['education'].value_counts()
```

Out[51]:

```
university.degree      12168
high.school             9515
basic.9y                6045
professional.course     5243
basic.4y                4176
basic.6y                2292
unknown                 1731
illiterate                18
Name: education, dtype: int64
```

In [52]:
```
data.education.replace(('basic.6y','basic.9y', 'basic.4y'), ('basic'), inplace=Tru
```

In [53]:
```
data = data.drop(['default'], axis=1)
#data = data.drop(columns=['day_of_week','month','contact','poutcome','pdays'],axis
```

In [54]:
```
data.y.replace(('yes', 'no'), (1, 0), inplace=True)
data.housing.replace(('yes', 'no'), (1, 0), inplace=True)
data.loan.replace(('yes', 'no'), (1, 0), inplace=True)
data
```

Out[54]:

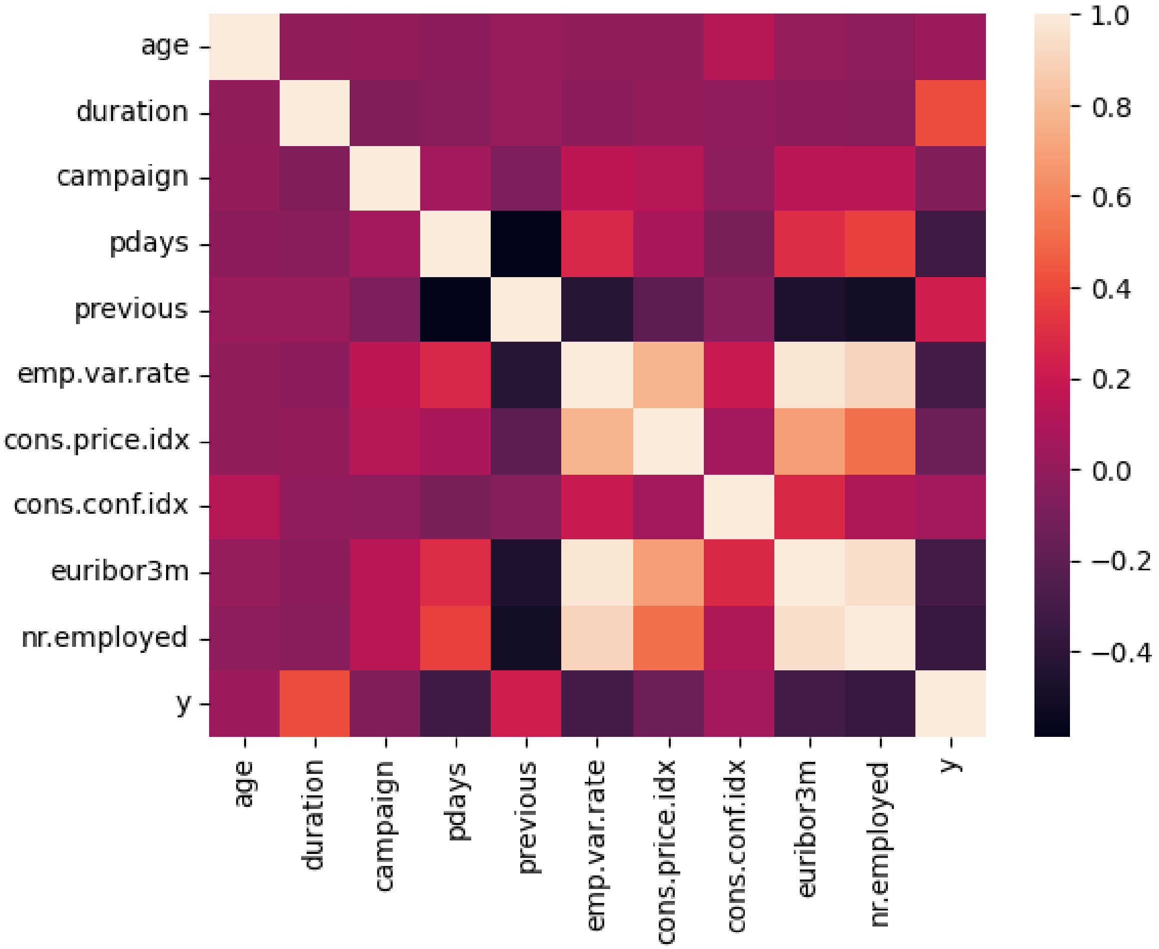
	age	job	marital	education	housing	loan	contact	month	day_of_wee
0	56	housemaid	married	basic	0	0	telephone	may	mo
1	57	services	married	high.school	0	0	telephone	may	mo
2	37	services	married	high.school	1	0	telephone	may	mo
3	40	admin.	married	basic	0	0	telephone	may	mo
4	56	services	married	high.school	0	1	telephone	may	mo
...	...	...	...	...	...	...	...	...	.
41183	73	retired	married	professional.course	1	0	cellular	nov	f
41184	46	blue-collar	married	professional.course	0	0	cellular	nov	f
41185	56	retired	married	university.degree	1	0	cellular	nov	f
41186	44	technician	married	professional.course	0	0	cellular	nov	f
41187	74	retired	married	professional.course	1	0	cellular	nov	f

41188 rows × 20 columns

In [57]:
```
import seaborn as sns

sns.heatmap(data.corr())
```

```
C:\Users\muham\AppData\Local\Temp\ipykernel_16888\458572859.py:3: FutureWarning: T
he default value of numeric_only in DataFrame.corr is deprecated. In a future vers
ion, it will default to False. Select only valid columns or specify the value of n
umeric_only to silence this warning.
  sns.heatmap(data.corr())
```

Out[57]: `<AxesSubplot: >`



In [58]:

```
df = pd.get_dummies(data)
df
```

Out[58]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	eur
0	56	261	1	999	0	1.1	93.994	-36.4	
1	57	149	1	999	0	1.1	93.994	-36.4	
2	37	226	1	999	0	1.1	93.994	-36.4	
3	40	151	1	999	0	1.1	93.994	-36.4	
4	56	307	1	999	0	1.1	93.994	-36.4	
...	...	...	...	...	...	...	...	...	
41183	73	334	1	999	0	-1.1	94.767	-50.8	
41184	46	383	1	999	0	-1.1	94.767	-50.8	
41185	56	189	2	999	0	-1.1	94.767	-50.8	
41186	44	442	1	999	0	-1.1	94.767	-50.8	
41187	74	239	3	999	1	-1.1	94.767	-50.8	

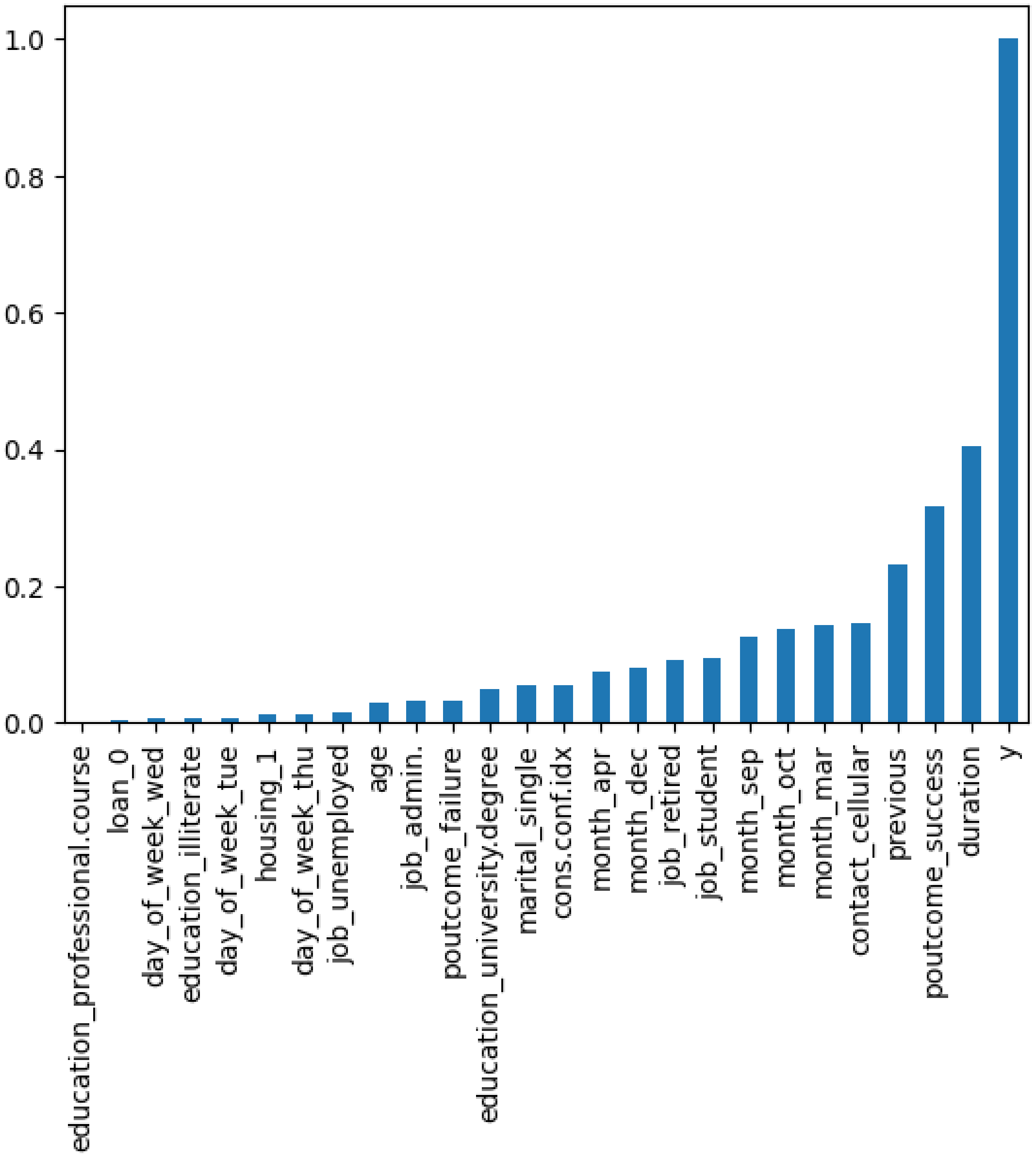
41188 rows × 59 columns

In [59]:

```
data=df.drop(columns=['job_unknown','marital_unknown','education_unknown'],axis=1)
```

In [60]:

```
cor = data.corr()['y']
chart = cor[cor >= 0].sort_values(ascending=True).plot(kind='bar')
```



In [61]:

```
X_data = data.drop(['y'], axis=1 )
```

In [62]:

```
Y_data = data['y']
Y_data.head()
```

Out[62]:

```
0    0
1    0
2    0
3    0
4    0
Name: y, dtype: int64
```

In [63]:

```
x_train, x_test, y_train, y_test = train_test_split(X_data, Y_data, random_state =
```

In [64]:

```
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [82]:

```
def scores(model, actual, predicted):
    c_matrix = confusion_matrix(actual, predicted)
    f1 = f1_score(actual, predicted)
    precision = precision_score(actual, predicted)
    recall = recall_score(actual, predicted)
    accuracy = accuracy_score(actual, predicted)
    model_name = str(model).split('(')[0]
    if model_name == 'GradientBoostingClassifier':
        model_name = 'GradientBoosting'
    print(f'{model_name}\t\t{round(f1*100, 2)}\t\t{round(precision*100, 2)}\t\t{rou
```

In [83]:

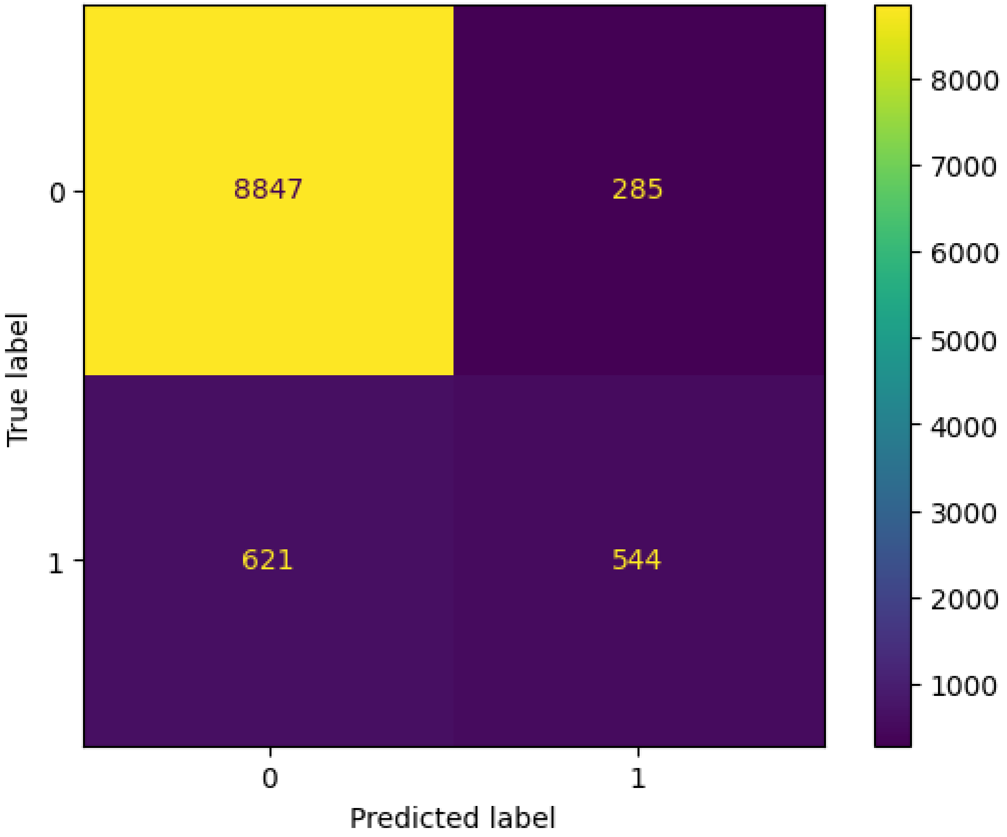
```
lr_model = LogisticRegression()
edt = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
gdt = DecisionTreeClassifier(criterion = 'gini', random_state = 0)
gb = GradientBoostingClassifier(learning_rate=0.01,random_state=1)
rf = RandomForestClassifier(n_estimators = 50)

models = [lr_model, edt, gdt, gb, rf]

print('Model\t\t\t\tF1 Score\tPrecision\tRecall\t\tAccuracy')
for model in models:
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    scores(model,y_test, y_pred)
    #print(confusion_matrix(y_test, y_pred))
```

```
Model                           F1 Score        Precision       Recall          Ac
curacy
LogisticRegression              52.11           66.09           43.0            9
1.06
DecisionTreeClassifier          52.71           51.96           53.48           8
9.14
DecisionTreeClassifier          53.68           52.32           55.11           8
9.24
GradientBoosting                25.02           81.9            14.76           8
9.99
RandomForestClassifier          54.74           66.3            46.61           9
1.28
```

In [78]:

```
rf = RandomForestClassifier(n_estimators = 50)
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
scores(model,y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

```
RandomForestClassifier  54.56           65.62           46.7            91.2
```


8847
285
621
544
0
1
True label
0
1
Predicted label
8000
7000
6000
5000
4000
3000
2000
1000

```
In [ ]:
```