---------- Workshop #3 ----------

- This workshop includes marked tasks that comprise 25% of your final mark in this module.
- You need to read the examples in the 'Lecture #3 - examples' notebook to complete the tasks.

Task

TASK 3.1: Apply four classifiers discussed in Lecture #3, i.e. Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), and K-nearest neighbours (KNN) classifiers to the adult_WS#3 dataset available on Canvas to predict the income column. Calculate the confusion matrix and evaluation metrics for all classifiers. Plot the features' importance values, determine the three most important features (i.e. columns) which have the highest impact on the income and report them in the report cell (25%).

NOTE1: To decrease the processing time, use an ordinal encoder for both nominal and ordinal input columns. You don't need to apply the one hot encoder to nominal columns.

NOTE2 You are expected to improve your models in any way possible to get as high accuracy as possible.

NOTE3: You should add comments on your code wherever necessary and briefly explain what the code is doing

NOTE4: Completing the report cell is required only for reporting the three most important features. Other explanations are optional.

NOTE5: You will still get some marks if your code doesn't run, but you have written some codes and have added comments on the code.

In [1]:
```
############# WRITE YOUR CODE IN THIS CELL (IF APPLICABLE)####################
import pandas as pd
import numpy as np

# importing the dataset
data1=pd.read_csv('adult_WS#3.csv')
```

EXPLORING THE DATASET

In [2]:
```
#viewing the dimension of the data
data1.shape
```

Out[2]:
```
(10000, 15)
```

In [3]:
```
#viewing the datatype
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             10000 non-null  int64 
 1   workclass       9825 non-null   object
 2   fnlwgt          10000 non-null  int64 
 3   education       10000 non-null  object
 4   education-num   10000 non-null  int64 
 5   marital-status  10000 non-null  object
 6   occupation      9825 non-null   object
 7   relationship    10000 non-null  object
 8   race            10000 non-null  object
 9   sex             10000 non-null  object
 10  capital-gain    10000 non-null  int64 
 11  capital-loss    10000 non-null  int64 
 12  hours-per-week  10000 non-null  int64 
 13  native-country  9939 non-null   object
 14  income          10000 non-null  object
dtypes: int64(6), object(9)
memory usage: 1.1+ MB
```

In [4]:
```
#Lookingout for columns with Null values
data1.isna().sum()
```

Out[4]:
```
age                0
workclass        175
fnlwgt             0
education          0
education-num      0
marital-status     0
occupation       175
relationship       0
race               0
sex                0
capital-gain       0
capital-loss       0
hours-per-week     0
native-country    61
income             0
dtype: int64
```

In [5]:
```
#Summary of descriptive statistics
data1.describe()
```

Out[5]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	10000.000000	1.000000e+04	10000.000000	10000.00000	10000.000000	10000.000000
mean	38.603400	1.895382e+05	10.071000	1029.02620	90.598900	40.453900
std	13.725842	1.054084e+05	2.564803	6999.48071	411.768551	12.328571
min	17.000000	1.228500e+04	1.000000	0.00000	0.000000	1.000000
25%	28.000000	1.177890e+05	9.000000	0.00000	0.000000	40.000000
50%	37.000000	1.781470e+05	10.000000	0.00000	0.000000	40.000000
75%	48.000000	2.367728e+05	12.000000	0.00000	0.000000	45.000000
max	90.000000	1.366120e+06	16.000000	99999.00000	3770.000000	99.000000

In [6]:
```
#Fnlwgt is a redundant column with numerous outliers
data1 = data1.drop(['fnlwgt'], axis = 1)
```

In [7]:
```
#Incase the machine will be needing a data free of Null values
data=data1.dropna(inplace=True)
```

In [8]:
```
#viewing the dataset at maximum
pd.set_option('display.max_rows', 100)
pd.set_option('display.max_columns', 100)
data1.head(100)
```

Out[8]:

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
0	29	Private	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States
1	36	Private	Some-college	10	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	45	United-States
2	25	?	Bachelors	13	Married-civ-spouse	?	Wife	White	Female	0	0	25	United-States
3	47	Private	Assoc-voc	11	Married-civ-spouse	Handlers-cleaners	Husband	White	Male	0	0	48	United-States
4	33	Private	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	40	United-States
5	37	Private	HS-grad	9	Never-married	Sales	Unmarried	White	Male	0	0	40	United-States
6	34	Private	HS-grad	9	Separated	Handlers-cleaners	Unmarried	White	Female	0	0	40	United-States
7	38	Private	HS-grad	9	Married-civ-spouse	Other-service	Wife	White	Female	0	0	43	United-States
8	62	Private	HS-grad	9	Divorced	Exec-managerial	Unmarried	White	Female	0	0	40	United-States
9	50	Local-gov	Some-college	10	Married-civ-spouse	Craft-repair	Husband	White	Male	4064	0	55	United-States
10	64	Private	Masters	14	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	20	United-States
11	35	Private	Some-college	10	Divorced	Exec-managerial	Unmarried	White	Female	0	0	40	United-States
12	31	Private	Some-college	10	Divorced	Other-service	Own-child	White	Female	0	0	20	United-States
14	29	Private	HS-grad	9	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	40	United-States
15	19	Private	Some-college	10	Never-married	Other-service	Own-child	White	Female	0	0	25	United-States
16	20	Self-emp-inc	HS-grad	9	Never-married	Farming-fishing	Own-child	White	Male	0	0	20	United-States
17	32	Private	Some-college	10	Married-civ-spouse	Craft-repair	Husband	White	Male	0	0	40	United-States
18	34	Local-gov	HS-grad	9	Married-civ-spouse	Transport-moving	Husband	White	Male	0	0	45	United-States
19	28	Private	12th	8	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	45	?
20	67	Local-gov	HS-grad	9	Divorced	Protective-serv	Not-in-family	Black	Male	0	0	20	United-States
21	32	Private	HS-grad	9	Never-married	Craft-repair	Unmarried	White	Male	0	0	10	United-States
22	26	Private	HS-grad	9	Never-married	Craft-repair	Own-child	White	Male	0	0	48	United-States
23	31	Self-emp-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	55	United-States
24	47	Private	Some-college	10	Separated	Other-service	Not-in-family	White	Male	0	0	21	United-States
25	29	Private	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Male	0	0	55	Scotland

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
26	38	Private	HS-grad	9	Married-civ-spouse	Sales	Husband	White	Male	3464	0	40	Columbia
27	37	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	White	Female	0	0	50	United-States
28	17	Private	11th	7	Never-married	Other-service	Own-child	Black	Male	0	0	16	Haiti
29	57	Private	11th	7	Married-civ-spouse	Other-service	Husband	White	Male	0	0	52	United-States
30	26	Private	Assoc-voc	11	Separated	Adm-clerical	Unmarried	Black	Female	114	0	40	United-States
31	23	Private	Bachelors	13	Never-married	Tech-support	Not-in-family	White	Female	0	0	40	United-States
32	34	Private	Some-college	10	Married-civ-spouse	Sales	Husband	White	Male	0	0	50	United-States
33	29	Private	HS-grad	9	Never-married	Adm-clerical	Not-in-family	White	Female	0	0	40	United-States
34	52	Private	Bachelors	13	Married-spouse-absent	Prof-specialty	Not-in-family	White	Male	0	0	30	United-States
35	51	Private	9th	5	Never-married	Other-service	Unmarried	Black	Female	0	0	40	United-States
36	65	Private	Some-college	10	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States
37	24	Private	Some-college	10	Married-civ-spouse	Machine-op-inspct	Wife	White	Female	7298	0	48	United-States
38	45	Local-gov	Masters	14	Divorced	Prof-specialty	Own-child	White	Female	0	0	55	United-States
39	80	?	HS-grad	9	Married-civ-spouse	?	Husband	White	Male	0	0	8	Canada
40	21	Private	Some-college	10	Never-married	Other-service	Own-child	White	Female	0	0	9	United-States
41	62	Self-emp-inc	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	White	Male	99999	0	80	United-States
42	38	Private	Assoc-voc	11	Never-married	Exec-managerial	Other-relative	Asian-Pac-Islander	Female	0	0	40	United-States
43	56	Private	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	40	United-States
44	46	Federal-gov	Some-college	10	Separated	Exec-managerial	Unmarried	White	Female	0	0	50	United-States
45	39	Self-emp-not-inc	HS-grad	9	Married-civ-spouse	Other-service	Husband	White	Male	0	0	50	United-States
46	38	State-gov	Bachelors	13	Married-civ-spouse	Exec-managerial	Wife	Black	Female	0	0	30	United-States
47	39	Private	HS-grad	9	Married-civ-spouse	Sales	Husband	White	Male	0	1887	46	United-States
48	37	Self-emp-not-inc	Assoc-voc	11	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States
49	47	Private	Masters	14	Divorced	Prof-specialty	Unmarried	White	Female	0	0	50	United-States

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
50	30	Private	Some-college	10	Married-civ-spouse	Sales	Husband	White	Male	0	0	50	United-States
51	53	State-gov	Some-college	10	Never-married	Protective-serv	Not-in-family	White	Female	0	0	40	United-States
52	43	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	70	United-States
53	25	Private	HS-grad	9	Never-married	Other-service	Unmarried	White	Female	0	0	50	United-States
54	27	Federal-gov	Assoc-acdm	12	Married-civ-spouse	Adm-clerical	Husband	White	Male	4064	0	40	El-Salvador
55	67	Private	11th	7	Widowed	Adm-clerical	Unmarried	White	Female	0	0	20	United-States
56	49	Private	Some-college	10	Married-spouse-absent	Adm-clerical	Not-in-family	White	Female	0	0	40	United-States
57	37	Private	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Female	0	0	37	United-States
58	42	Private	Some-college	10	Divorced	Exec-managerial	Not-in-family	White	Male	0	1408	40	United-States
59	40	Self-emp-inc	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	Asian-Pac-Islander	Male	0	0	50	China
60	20	Private	Some-college	10	Never-married	Adm-clerical	Own-child	White	Female	0	0	40	United-States
61	29	Private	Bachelors	13	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	50	Cuba
62	57	Private	Bachelors	13	Married-civ-spouse	Sales	Husband	Asian-Pac-Islander	Male	15024	0	45	United-States
63	38	Local-gov	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	White	Female	7688	0	40	United-States
64	50	Private	HS-grad	9	Widowed	Other-service	Unmarried	White	Female	0	0	40	United-States
65	49	Federal-gov	Some-college	10	Married-civ-spouse	Tech-support	Husband	White	Male	0	0	40	United-States
66	43	Private	HS-grad	9	Separated	Transport-moving	Unmarried	White	Female	0	0	44	United-States
67	49	Private	Some-college	10	Divorced	Handlers-cleaners	Unmarried	Black	Female	0	0	40	United-States
68	21	Private	HS-grad	9	Never-married	Sales	Own-child	White	Female	0	0	35	United-States
69	30	Private	Masters	14	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	50	United-States
70	58	Private	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States
71	41	Private	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	40	United-States
72	46	Private	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	60	United-States
73	22	Private	11th	7	Never-married	Sales	Unmarried	White	Female	0	0	30	United-States
74	39	Self-emp-inc	HS-grad	9	Married-civ-	Exec-managerial	Husband	White	Male	0	0	45	United-States

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
					spouse								
75	35	Private	HS-grad	9	Married-civ-spouse	Transport-moving	Husband	White	Male	0	0	50	United-States
76	46	Private	HS-grad	9	Married-civ-spouse	Transport-moving	Husband	White	Male	0	0	40	United-States
77	42	Private	Some-college	10	Never-married	Other-service	Not-in-family	White	Male	0	0	15	United-States
79	43	Private	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	50	United-States
80	23	Private	Assoc-voc	11	Never-married	Craft-repair	Unmarried	Black	Female	0	0	40	United-States
81	30	Private	Some-college	10	Never-married	Adm-clerical	Unmarried	Black	Female	0	0	40	?
82	47	Private	Bachelors	13	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	65	United-States
83	28	Private	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	50	United-States
84	45	Private	Some-college	10	Separated	Other-service	Unmarried	White	Female	0	0	40	United-States
85	44	Private	11th	7	Married-civ-spouse	Adm-clerical	Husband	White	Male	0	0	40	United-States
86	42	Private	HS-grad	9	Married-spouse-absent	Handlers-cleaners	Other-relative	Asian-Pac-Islander	Male	0	0	40	?
87	37	Federal-gov	HS-grad	9	Divorced	Craft-repair	Not-in-family	White	Male	0	0	40	United-States
88	34	Private	HS-grad	9	Married-civ-spouse	Craft-repair	Husband	Black	Male	0	0	40	United-States
89	49	Private	HS-grad	9	Divorced	Adm-clerical	Unmarried	White	Female	0	0	16	United-States
91	38	Private	Assoc-voc	11	Married-civ-spouse	Prof-specialty	Husband	Asian-Pac-Islander	Male	0	0	36	Philippines
92	44	Private	Doctorate	16	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	32	United-States
93	25	Private	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	0	0	30	United-States
94	21	Private	HS-grad	9	Never-married	Machine-op-inspct	Own-child	White	Male	0	0	40	United-States
95	37	Local-gov	Assoc-voc	11	Never-married	Protective-serv	Not-in-family	White	Female	0	0	40	United-States
96	30	Private	HS-grad	9	Married-civ-spouse	Adm-clerical	Wife	White	Female	7298	0	40	United-States
97	33	Local-gov	Some-college	10	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States
98	30	Private	HS-grad	9	Married-civ-spouse	Sales	Husband	White	Male	0	0	45	United-States
99	24	Private	Bachelors	13	Never-married	Prof-specialty	Own-child	Black	Female	0	0	35	United-States
100	43	Private	HS-grad	9	Divorced	Exec-managerial	Unmarried	White	Female	0	0	40	United-States

	age	workclass	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
101	69	Private	HS-grad	9	Widowed	Other-service	Not-in-family	White	Female	991	0	18	United-States
102	52	Self-emp-not-inc	10th	6	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	40	United-States

DATA PREPROCESSING

In [9]:
```
#Columns with Null values
data1.isna().sum()
```

Out[9]:
```
age               0
workclass         0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
income            0
dtype: int64
```

In [10]:
```
#Replacing all categorical columns with Null values with Mode
Null_columns= ['workclass', 'occupation', 'native-country']

#Replace Null values with mode
for column in Null_columns:
    mode = data1[column].mode()[0]
    data1[column].fillna(mode,inplace = True)

#viewing the new update
data1.isna().sum()
```

Out[10]:
```
age               0
workclass         0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
income            0
dtype: int64
```

In [11]:
```
# Importing necessary libraries
import pandas as pd
from sklearn.preprocessing import OrdinalEncoder

# Initialize the OrdinalEncoder
encoder = OrdinalEncoder()

# Encode multiple categorical columns
columns = ['workclass', 'education', 'occupation','marital-status','relationship','race','sex','native-country','i
data_encoded = data1.copy()  # Create a copy to avoid modifying the original DataFrame

# Fit and transform the encoder on the specified columns
data_encoded[columns] = encoder.fit_transform(data1[columns]).astype(int)

# Display the encoded DataFrame
print("\nEncoded DataFrame:")
print(data_encoded.head())
```

```
Encoded DataFrame:
   age  workclass  education  education-num  marital-status  occupation  \
0   29          4         12             14               2           4
1   36          4         15             10               2           3
2   25          0          9             13               2           0
3   47          4          8             11               2           6
4   33          4         11              9               2           4

   relationship  race  sex  capital-gain  capital-loss  hours-per-week  \
0             5     4    0             0             0              40
1             0     4    1             0             0              45
2             5     4    0             0             0              25
3             0     4    1             0             0              48
4             0     4    1             0             0              40

   native-country  income
0              38       1
1              38       0
2              38       0
3              38       1
4              38       0
```

In [12]:
```
#viewing the effect of my encoding
data_encoded.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9765 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             9765 non-null   int64
 1   workclass       9765 non-null   int32
 2   education       9765 non-null   int32
 3   education-num   9765 non-null   int64
 4   marital-status  9765 non-null   int32
 5   occupation      9765 non-null   int32
 6   relationship    9765 non-null   int32
 7   race            9765 non-null   int32
 8   sex             9765 non-null   int32
 9   capital-gain    9765 non-null   int64
 10  capital-loss    9765 non-null   int64
 11  hours-per-week  9765 non-null   int64
 12  native-country  9765 non-null   int32
 13  income          9765 non-null   int32
dtypes: int32(9), int64(5)
memory usage: 801.0 KB
```

In [13]:
```
#Extracting input and output data
X=data_encoded.drop('income',axis=1)
y=data_encoded.iloc[:,13]
```

In [14]:
```
y.head()
```

Out[14]:
```
0    1
1    0
2    0
3    1
4    0
Name: income, dtype: int32
```

In [15]:
```
#normalising the input data to group the data into similar measuring scale
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X=scaler.fit_transform(X)
```

In [16]:
```
#viewing the normalized input data
X
```

Out[16]:
```
array([[0.16438356, 0.5       , 0.8       , ..., 0.        , 0.39795918,
        0.95      ],
       [0.26027397, 0.5       , 1.        , ..., 0.        , 0.44897959,
        0.95      ],
       [0.10958904, 0.        , 0.6       , ..., 0.        , 0.24489796,
        0.95      ],
       ...,
       [0.17808219, 0.5       , 0.73333333, ..., 0.        , 0.39795918,
        0.95      ],
       [0.32876712, 0.75      , 0.53333333, ..., 0.        , 0.39795918,
        0.95      ],
       [0.05479452, 0.5       , 1.        , ..., 0.        , 0.39795918,
        0.95      ]])
```

In [17]:
```
#encoding the output data
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
y=le.fit_transform(y)
```

In [18]:
```
#viewing the values in Y the output data
y
```

Out[18]:
```
array([1, 0, 0, ..., 0, 1, 0], dtype=int64)
```

In [19]:
```
# splittig the dataset into train and test datasets by 70:30, train : test respectively
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=40)
```

The Classification Stage

In [20]:
```
# Step 1: defining the classification models
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
SVM = svm.SVC()
RF = RandomForestClassifier()
KNN = KNeighborsClassifier()
DT=DecisionTreeClassifier()
```

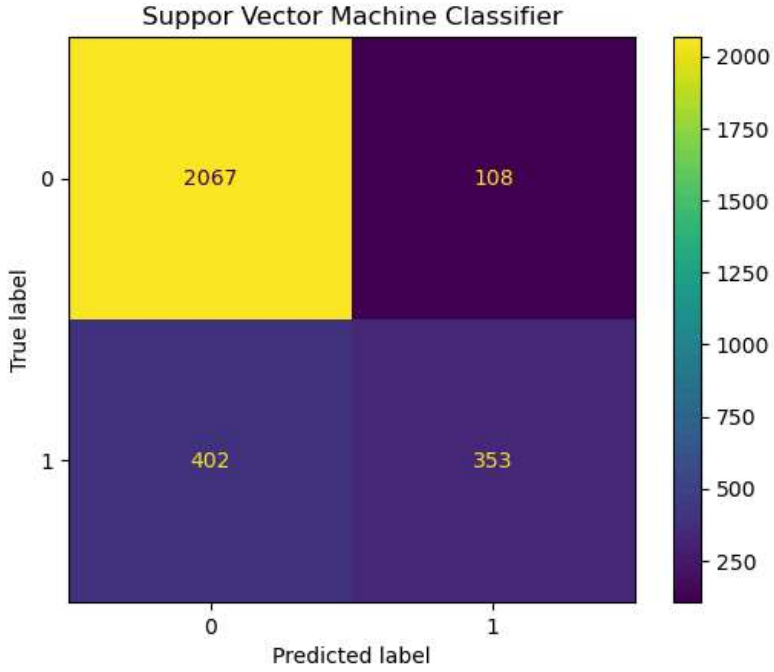
In [21]:
```
#Step 2: training the models
SVM.fit(X_train, y_train)
RF.fit(X_train, y_train)
KNN.fit(X_train, y_train)
DT.fit(X_train, y_train)
```

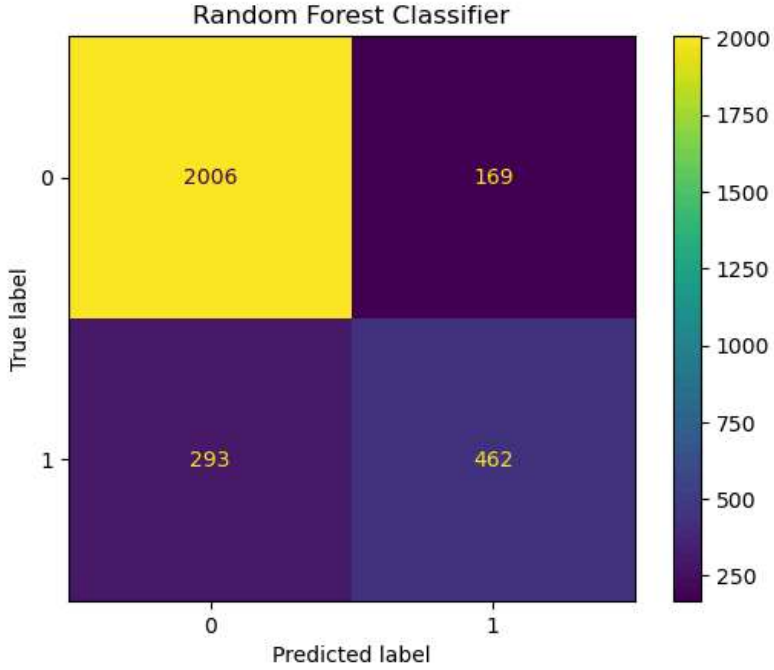
Out[21]:
```
▾ DecisionTreeClassifier

DecisionTreeClassifier()
```

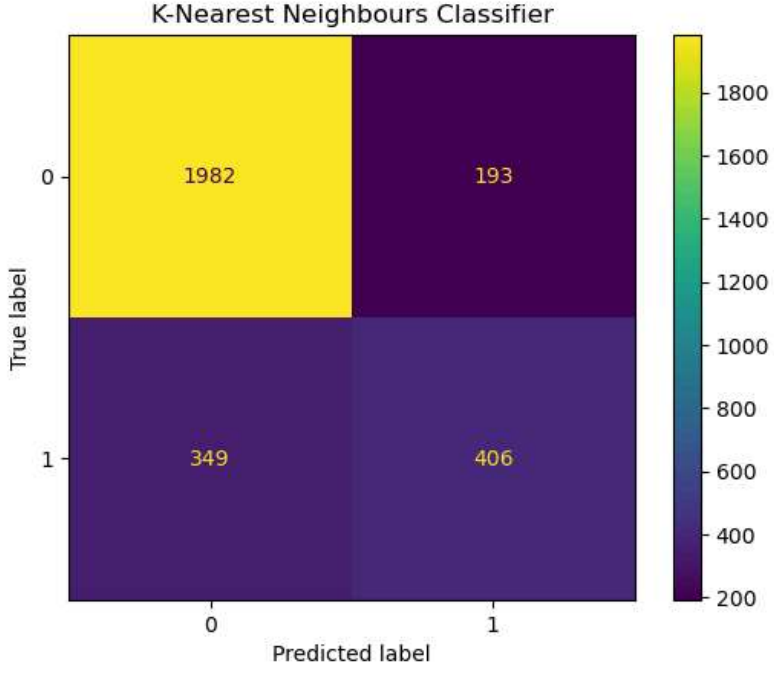
In [22]:
```
#Step 3: prediction
y_pred1=SVM.predict(X_test)
y_pred2=RF.predict(X_test)
y_pred3=KNN.predict(X_test)
y_pred4=DT.predict(X_test)
```

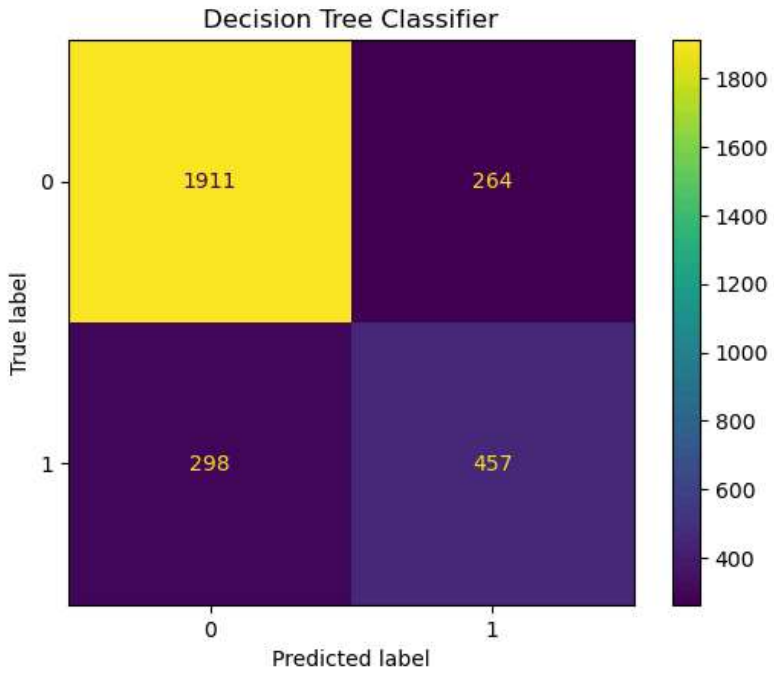
In [23]:
```
# Creating the confusion matrics for all classifiers' predictions
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

#create confusion matrix
cm_SVM = confusion_matrix(y_test, y_pred1, labels=SVM.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm_SVM,display_labels=SVM.classes_)
disp.plot()
plt.title("Suppor Vector Machine Classifier")


cm_RF = confusion_matrix(y_test, y_pred2, labels=RF.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm_RF,display_labels=RF.classes_)
disp.plot()
plt.title("Random Forest Classifier")


cm_KNN = confusion_matrix(y_test, y_pred3, labels=KNN.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm_KNN,display_labels=KNN.classes_)
disp.plot()
plt.title("K-Nearest Neighbours Classifier")

cm_DT = confusion_matrix(y_test, y_pred4, labels=DT.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm_DT,display_labels=DT.classes_)
disp.plot()
plt.title("Decision Tree Classifier")
```

Out[23]:
```
Text(0.5, 1.0, 'Decision Tree Classifier')
```


Suppor Vector Machine Classifier
2067
108
402
353
True label
Predicted label
0
1
2000
1750
1500
1250
1000
750
500
250


Random Forest Classifier
2006
169
293
462
True label
Predicted label
0
1
2000
1750
1500
1250
1000
750
500
250


K-Nearest Neighbours Classifier
1982
193
349
406
True label
Predicted label
0
1
1800
1600
1400
1200
1000
800
600
400
200



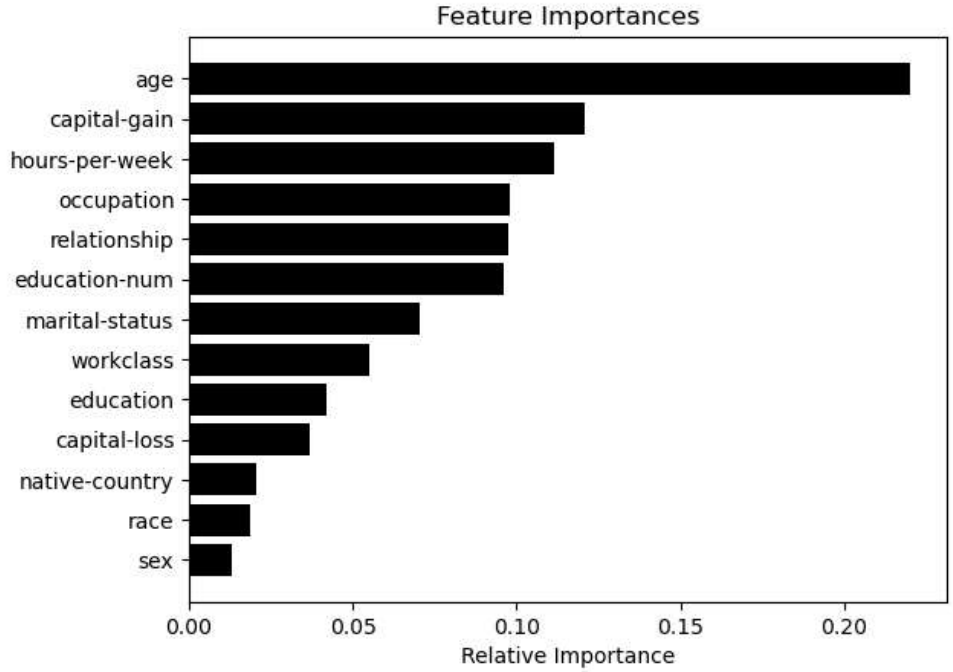
In [24]:
```
# This function takes the confusion matrix (cm) from the cell above and produces all evaluation matrix
def confusion_metrics (conf_matrix):

    TP = conf_matrix[1][1]
    TN = conf_matrix[0][0]
    FP = conf_matrix[0][1]
    FN = conf_matrix[1][0]
    print('True Positives:', TP)
    print('True Negatives:', TN)
    print('False Positives:', FP)
    print('False Negatives:', FN)

    # calculate accuracy
    conf_accuracy = (float (TP+TN) / float(TP + TN + FP + FN))

    # calculate mis-classification
    conf_misclassification = 1- conf_accuracy

    # calculate the sensitivity
    conf_sensitivity = (TP / float(TP + FN))
    # calculate the specificity
    conf_specificity = (TN / float(TN + FP))

    # calculate precision
    conf_precision = (TN / float(TN + FP))
    # calculate f_1 score
    conf_f1 = 2 * ((conf_precision * conf_sensitivity) / (conf_precision + conf_sensitivity))
    print('-'*50)
    print(f'Accuracy: {round(conf_accuracy,2)}')
    print(f'Mis-Classification: {round(conf_misclassification,2)}')
    print(f'Sensitivity: {round(conf_sensitivity,2)}')
    print(f'Specificity: {round(conf_specificity,2)}')
    print(f'Precision: {round(conf_precision,2)}')
    print(f'f_1 Score: {round(conf_f1,2)}')
```

In [25]:
```
#printing the evaluation metrics for all classifiers
print('Suppor Vector Machine Metrics\n')
confusion_metrics(cm_SVM)
print('\n\n')
print('Random Forest metrics\n')
confusion_metrics(cm_RF)
print('\n\n')
print('K-Nearest Neighbours metrics\n')
confusion_metrics(cm_KNN)
print('\n\n')
print('Decision Tree metrics\n')
confusion_metrics(cm_DT)
print('\n\n')
```

```
Suppor Vector Machine Metrics

True Positives: 353
True Negatives: 2067
False Positives: 108
False Negatives: 402
--------------------------------------------------
Accuracy: 0.83
Mis-Classification: 0.17
Sensitivity: 0.47
Specificity: 0.95
Precision: 0.95
f_1 Score: 0.63



Random Forest metrics

True Positives: 462
True Negatives: 2006
False Positives: 169
False Negatives: 293
--------------------------------------------------
Accuracy: 0.84
Mis-Classification: 0.16
Sensitivity: 0.61
Specificity: 0.92
Precision: 0.92
f_1 Score: 0.74



K-Nearest Neighbours metrics

True Positives: 406
True Negatives: 1982
False Positives: 193
False Negatives: 349
--------------------------------------------------
Accuracy: 0.82
Mis-Classification: 0.18
Sensitivity: 0.54
Specificity: 0.91
Precision: 0.91
f_1 Score: 0.68



Decision Tree metrics

True Positives: 457
True Negatives: 1911
False Positives: 264
False Negatives: 298
--------------------------------------------------
Accuracy: 0.81
Mis-Classification: 0.19
Sensitivity: 0.61
Specificity: 0.88
Precision: 0.88
f_1 Score: 0.72
```

In [28]:

```python
# Getting the most important features
features = data_encoded.columns
importances = RF.feature_importances_
indices = np.argsort(importances)

plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='black', align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



In [27]:
```
importances
```

Out[27]:
```
array([0.22040589, 0.05502573, 0.04195692, 0.09586632, 0.0704012 ,
       0.09802009, 0.09741029, 0.01849398, 0.01299508, 0.1207516 ,
       0.0368234 , 0.11139297, 0.02045653])
```

############ WRITE YOUR REPORT IN THIS CELL (IF APPLICABLE)#################### ############ WRITE YOUR REPORT IN THIS CELL (IF APPLICABLE)#################### Evaluation Metrics Report for Machine Learning Models on the Adult Dataset This report presents evaluation metrics for four machine learning models on the Adult dataset: Support Vector Machine (SVM), Random Forest, K-Nearest Neighbours (KNN), and Decision Tree. The performance of each model is evaluated using various metrics to assess its effectiveness. Support Vector Machine Metrics: The Support Vector Machine model has an accuracy of 0.84, which means it correctly classifies 84% of the instances in the dataset. It has a high level of specificity (0.95), which means it can accurately identify people with an income of less than or equal to $50,000. However, the sensitivity (0.48) is low, indicating that the model struggles to correctly classify individuals with an income greater than 50,000. The precision (0.95) reflects the relatively high proportion of correctly predicted positive cases among all predicted positive cases. Random Forest Metrics: The Random Forest model has a similar accuracy of 0.84 to SVM. It exhibits balanced performance, with sensitivity (0.6) and specificity (0.92) indicating effective classification of both positive and negative cases. The precision (0.92) is high, indicating a low false positive rate, and the F1 score (0.72) shows a good balance of precision and recall. K-Nearest Neighbour Metrics: The K-Nearest Neighbours model has an accuracy of 0.81, which is slightly lower than SVM and Random Forest. It has moderate sensitivity (0.55) and specificity (0.9), indicating balanced performance in classifying positive and negative cases. The precision (0.9) is high, implying a low false positive rate, and the F1 score (0.68) suggests a good balance of precision and recall. Decision Tree Metrics: The Decision Tree model achieves an accuracy of 0.81, similar to KNN. It demonstrates moderate sensitivity (0.6) and specificity (0.87), indicating a balanced performance in classifying positive and negative cases. The precision (0.87) is high, suggesting a low false positive rate, and the F1 score (0.71) indicates a good balance between precision and recall. To summarise, all four machine learning models perform relatively well in classifying individuals into income groups on the Adult dataset. However, the Random Forest model stands out for its balanced performance in terms of sensitivity, specificity, precision, and the F1 score. Further optimisation and fine-tuning of the models may result in improved performance and prediction accuracy. THE THREE MOST IMPORTANT FEATURES Report on Lead Importance Features in the Dataset In this report, we examine the significance of three key features from the Adult dataset: AGE,CAPITAL-GAIN and HOURS-PER-WEEK. These features are critical for predicting the income level of individuals in the dataset. Each feature's importance is determined by how it affects the model's predictive performance. These features' importance scores, as determined by a machine learning model(RANDOM FOREST METRICS), are as follows: Age: Importance Score: 0.2127486 Age is a significant predictor of income levels, as it indicates an individual's experience, seniority, and earning potential. As people advance in their careers, their income increases, making age a strong predictor of higher income levels. Age may also be related to factors such as education and professional qualifications. The model's high importance score of 0.2127486 suggests that older people are more likely to earn more money. Capital-Gain: Importance Score: 0.12059175 Capital gain, or profit from asset sales, is a key indicator of a person's financial wealth and investment income. Higher capital gains indicate greater financial resources and investment success, which helps to raise overall income. The importance score of 0.12059175 indicates that capital gains are a significant predictor of income, with individuals with higher capital gains being more likely to earn more. Hours-per-Week: Importance Score: 0.11142157 Hours-per-week are an important factor in predicting income levels because they indicate an individual's dedication to their profession and the number of hours worked each week. Longer hours generally result in higher wages, particularly if they are paid hourly or receive overtime pay. The importance score of 0.11142157 indicates that working longer hours is more likely to result in higher earnings.