Problem Statement 1: - Mandatory

Explore the given data set with EDA techniques and build a suitable model for predicting whether the salary of the person is >50k or not and visualize the results. NOTE: the algorithm used for the model must be built from the scratch.

PROBLEM STATEMENT AND ANALYSIS:

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

A support vector machine (SVM) is machine learning algorithm that analyzes data for classification and regression analysis. SVM is a supervised learning method that looks at data and sorts it into one of two categories. An SVM outputs a map of the sorted data with the margins between the two as far apart as possible. SVMs are used in text categorization, image classification, handwriting recognition and in the sciences.A support vector machine is also known as a support vector network (SVN).

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

CODE:

```
#DATA PREPROCESSING
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
dt = pd.read_csv('train.csv')
dt.head()
dt.info()
dt.shape
dt.describe()
```

```
dt.isnull().sum()

train=dt.dropna()
train.isnull().sum()

sns.heatmap(train.corr(),cmap='coolwarm')

#encoding
train.columns = ['age', 'workclass', 'fnlwgt', 'education', 'education.num','marital.status','occupation','relationship','race','sex','captial.gain','capital.loss','hours.per.week','native.country','target']

from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
train['sex']=le.fit_transform(train['sex'])
train['marital.status']=le.fit_transform(train['marital.status'])
train['education']=le.fit_transform(train['education'])
train['relationship']=le.fit_transform(train['relationship'])
train['race']=le.fit_transform(train['race'])
train['occupation']=le.fit_transform(train['occupation'])
train['workclass']=le.fit_transform(train['workclass'])
train['native.country']=le.fit_transform(train['native.country'])
train.head()

dd= pd.read_csv('test.csv')

dd.head()
dd.shape

dd.info()
dd.describe()

dd.isnull().sum()

test=dt.dropna()
test.isnull().sum()
```

```
#encoding
test.columns = ['age', 'workclass', 'fnlwgt', 'education', 'education.num','marital.status','occupation'
,'relationship','race','sex','captial.gain','capital.loss','hours.per.week','native.country','target']

from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
test['sex']=le.fit_transform(test['sex'])
test['marital.status']=le.fit_transform(test['marital.status'])
test['education']=le.fit_transform(test['education'])
test['relationship']=le.fit_transform(test['relationship'])
test['race']=le.fit_transform(test['race'])
test['occupation']=le.fit_transform(test['occupation'])
test['workclass']=le.fit_transform(test['workclass'])
test['native.country']=le.fit_transform(test['native.country'])

test.head()



from sklearn.model_selection import train_test_split
X = train[['age', 'workclass', 'fnlwgt', 'education', 'education.num','marital.status','occupation','rela
tionship','race','sex','captial.gain','capital.loss','hours.per.week','native.country']]
y = train['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)


#logestic regression
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
valid_predictions = logmodel.predict(X_test)
from sklearn.metrics import classification_report
print(classification_report(y_test,valid_predictions))


#svm
from sklearn.svm import SVC
svc_model = SVC()
svc_model.fit(X_train,y_train)
svm_predictions = svc_model.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
```

```
print(confusion_matrix(y_test,svm_predictions))
print(classification_report(y_test,svm_predictions))

#random forest

from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=600)
rfc.fit(X_train,y_train)
rf_predictions = rfc.predict(X_test)
print(classification_report(y_test,rf_predictions))


#testing
rfc = RandomForestClassifier(n_estimators=600)
rfc.fit(X,y)
final_predictions = rfc.predict(test.drop(['target'],axis=1))
print(confusion_matrix(test['target'], final_predictions))
```

<u>OUTPUT:</u>

<u>INFO</u>

```
[ ]  dt.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       30725 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64
 5   marital.status  32561 non-null  object
 6   occupation      30718 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64
 11  capital.loss    32561 non-null  int64
 12  hours.per.week  32561 non-null  int64
 13  native.country  31978 non-null  object
 14  target          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

<u>DESCRIBE</u>

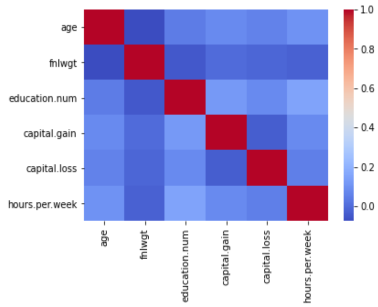
```
[ ] dt.describe()
```

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

HEATMAP

```
[ ] sns.heatmap(train.corr(),cmap='coolwarm')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc6a9fb3610>
```



ENCODED TRAIN VALUE

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	captial.gain	capital.loss	hours.per.week	native.cou
0	39	5	77516	9	13	4	0	1	4	1	2174	0	40	
1	50	4	83311	9	13	2	3	0	4	1	0	0	13	
2	38	2	215646	11	9	0	5	1	4	1	0	0	40	
3	53	2	234721	1	7	2	5	0	2	1	0	0	40	

INFO

```
dd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16281 entries, 0 to 16280
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             16281 non-null  int64
 1   workclass       15318 non-null  object
 2   fnlwgt          16281 non-null  int64
 3   education       16281 non-null  object
 4   education.num   16281 non-null  int64
 5   marital.status  16281 non-null  object
 6   occupation      15315 non-null  object
 7   relationship    16281 non-null  object
 8   race            16281 non-null  object
 9   sex             16281 non-null  object
 10  capital.gain    16281 non-null  int64
 11  capital.loss    16281 non-null  int64
 12  hours.per.week  16281 non-null  int64
 13  native.country  16007 non-null  object
 14  target          16281 non-null  object
dtypes: int64(6), object(9)
memory usage: 1.9+ MB
```

DESCRIBE

```
dd.describe()
```

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
count	16281.000000	1.628100e+04	16281.000000	16281.000000	16281.000000	16281.000000
mean	38.767459	1.894357e+05	10.072907	1081.905104	87.899269	40.392236
std	13.849187	1.057149e+05	2.567545	7583.935968	403.105286	12.479332
min	17.000000	1.349200e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.167360e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.778310e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.383840e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.490400e+06	16.000000	99999.000000	3770.000000	99.000000

ENCODED TEST VALUE

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	captial.gain	capital.loss	hours.per.week	native.cou
0	39	5	77516	9	13	4	0	1	4	1	2174	0	40	
1	50	4	83311	9	13	2	3	0	4	1	0	0	13	
2	38	2	215646	11	9	0	5	1	4	1	0	0	40	
3	53	2	234721	1	7	2	5	0	2	1	0	0	40	

LOGISTIC REGRESSION

```
print(classification_report(y_test,valid_predictions))

              precision    recall  f1-score   support

       <=50K       0.80      0.95      0.87      7479
        >50K       0.64      0.29      0.40      2475

    accuracy                           0.78      9954
   macro avg       0.72      0.62      0.63      9954
weighted avg       0.76      0.78      0.75      9954
```

SVM

```
[ ] print(classification_report(y_test,svm_predictions))

              precision    recall  f1-score   support

       <=50K       0.78      1.00      0.88      7479
        >50K       0.98      0.15      0.26      2475

    accuracy                           0.79      9954
   macro avg       0.88      0.57      0.57      9954
weighted avg       0.83      0.79      0.72      9954
```

RANDOM FOREST

```
print(classification_report(y_test,rf_predictions))

              precision    recall  f1-score   support

       <=50K       0.89      0.92      0.91      7479
        >50K       0.74      0.64      0.69      2475

    accuracy                           0.85      9954
   macro avg       0.81      0.78      0.80      9954
weighted avg       0.85      0.85      0.85      9954
```

CONCLUSION:

Comparing the accuracy of the different models, Random forest is the best.
It gives accuracy of about 85% than compared to other to model.so random forest is better suited for this type of dataset.the salary of the person >50 or not are also shown by each type of model.