Task 1

# \*\*1. Importing Necessary Libraries\*\*

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

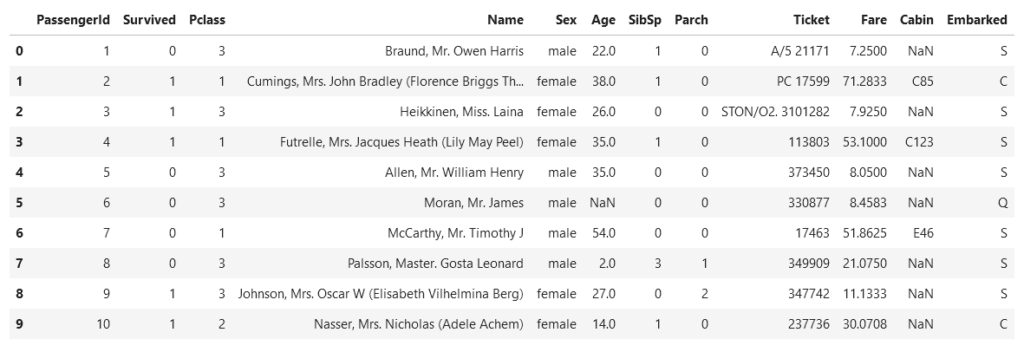
from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn import tree,svm

from sklearn.metrics import accuracy\_score

train\_data = pd.read\_csv('/kaggle/input/titanic/train.csv') # Printing first 10 rows of the dataset train\_data.head(10)

print('The shape of our training set: %s passengers and %s features'%(train\_data.shape[0],train\_data.shape[1]))

train\_data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 891 entries, 0 to 890

Data columns (total 12 columns):

Column Non-Null Count Dtype

--- ------ -------------- -----

0 PassengerId 891 non-null int64

1 Survived 891 non-null int64

2 Pclass 891 non-null int64

3 Name 891 non-null object

4 Sex 891 non-null object

5 Age 714 non-null float64

6 SibSp 891 non-null int64

7 Parch 891 non-null int64

8 Ticket 891 non-null object

9 Fare 891 non-null float64

10 Cabin 204 non-null object

11 Embarked 889 non-null object

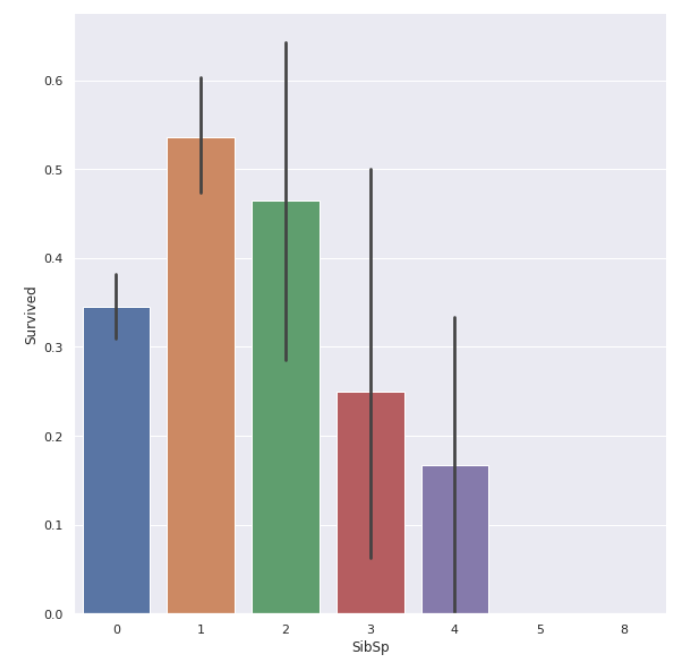
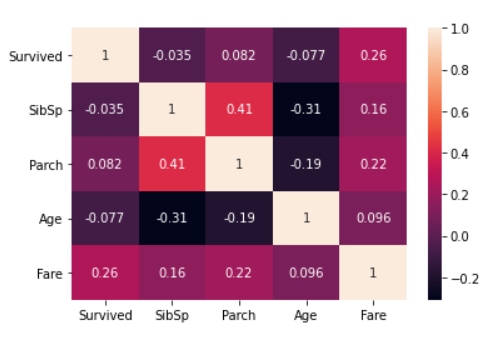
dtypes: float64(2), int64(5), object(5)

memory usage: 83.7+ KB

train\_data.isnull().sum()

PassengerId 0 Survived 0 Pclass 0 Name 0 Sex 0 Age 177 SibSp 0 Parch 0 Ticket 0 Fare 0 Cabin 687 Embarked 2 dtype: int64

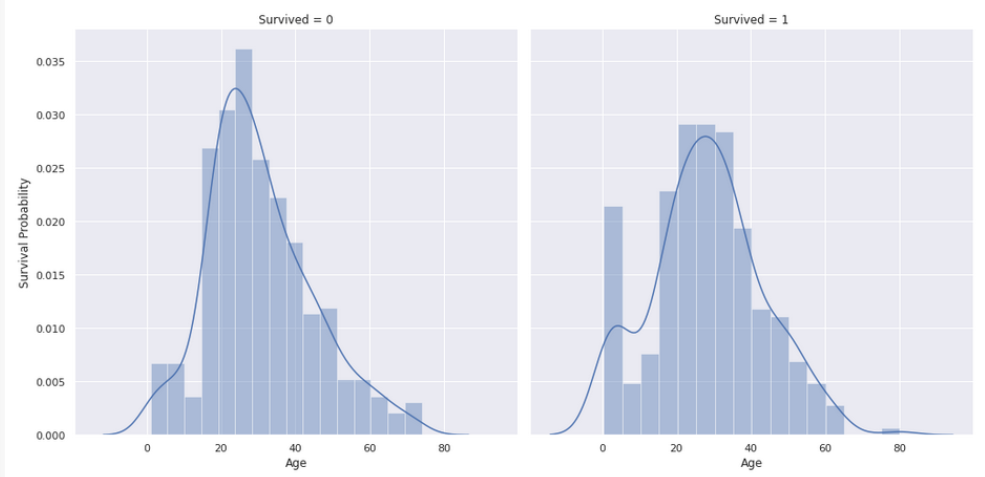
heatmap = sns.heatmap(train\_data[["Survived", "SibSp", "Parch", "Age", "Fare"]].corr(), annot = True) sns.set(rc={'figure.figsize':(12,10)})



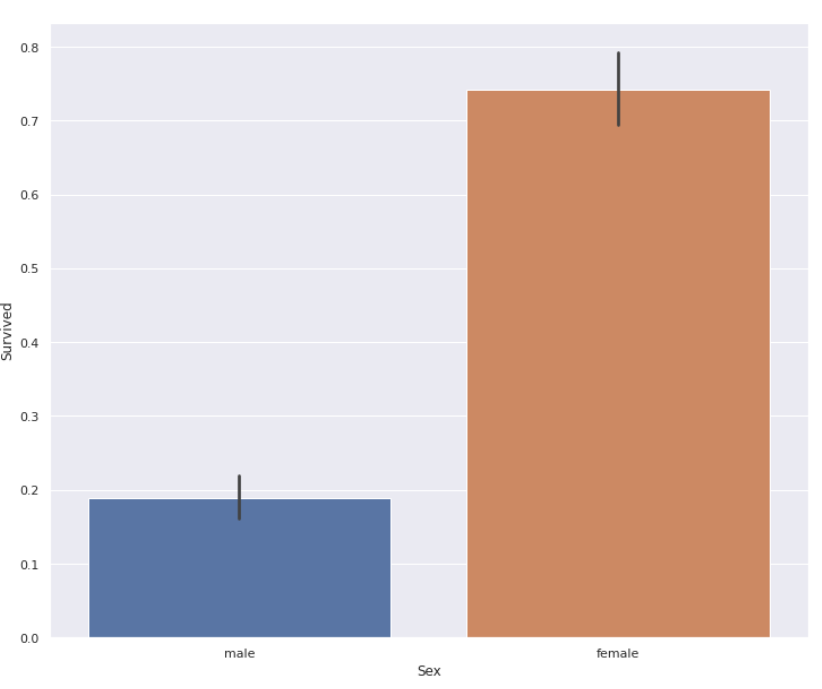
ageplot = sns.FacetGrid(train\_data, col="Survived", height = 7)

ageplot = ageplot.map(sns.distplot, "Age")

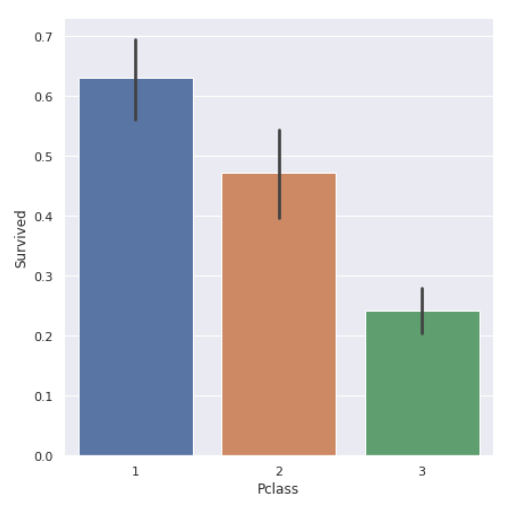
ageplot = ageplot.set\_ylabels("Survival Probability")



sexplot = sns.barplot(x="Sex", y="Survived", data=train\_data)

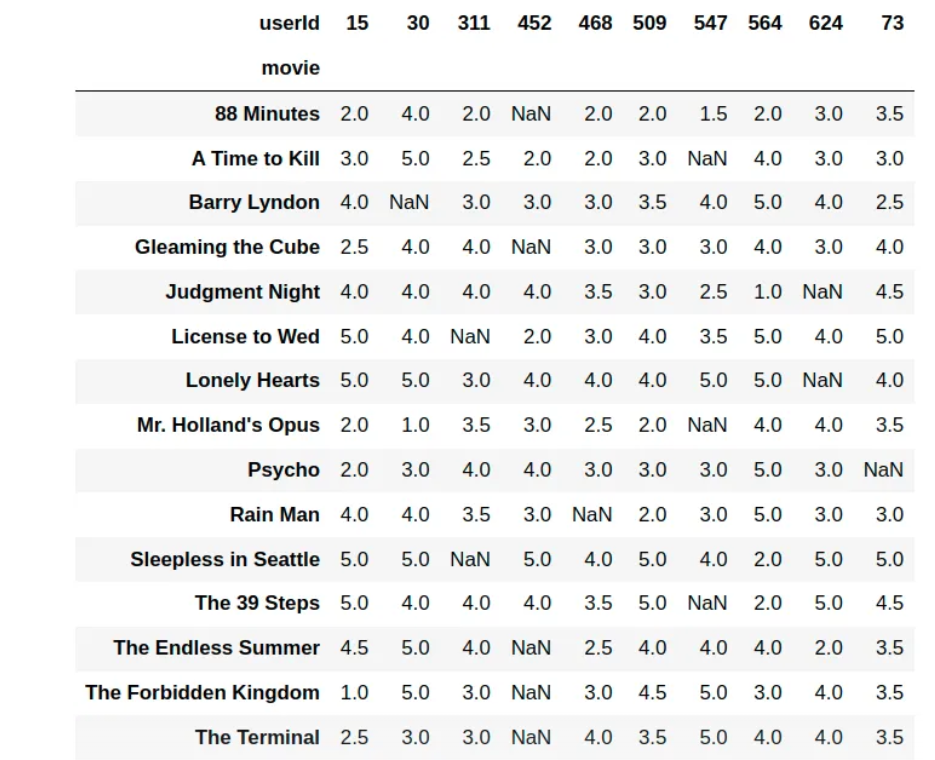


pclassplot = sns.catplot(x = "Pclass", y="Survived", data = train\_data, kind="bar", height = 6)



Task 2

|  |
| --- |
| import numpy as np |
|  | import pandas as pd |
|  | ratings\_df = pd.read\_csv('small\_movie\_ratings.csv', index\_col=0) |
|  | ratings\_df |



import io

import os

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import json *#converting JSON to lists for dataframe*import warningswarnings.filterwarnings('ignore')

import base64

import codecsfrom IPython.display

import HTML

%matplotlib inline

movie1 = pd.read\_csv("../input/tmdb-movie-metadata/tmdb\_5000\_movies.csv")

movie2 = pd.read\_csv("../input/tmdb-movie-metadata/tmdb\_5000\_credits.csv")

movies = movie1.merge(movie2,left\_on='id',right\_on='movie\_id',how='left')

counts = movies[(movies['vote\_average']==0)]['vote\_count'] *# get vote counts for all movies that have a rating of 0.0*

print("Unique vote counts for movies with 0.0 rating")for u **in** set(counts):

print(u)

Unique vote counts for movies with 0.0 rating

0

1

movies = movies[(movies['vote\_average']!=0)]

Quick look at movies

movies.sample(5)

| budget | genres | homepage | id | keywords | original\_language | original\_title | overview | popularity | production\_companies | ... | spoken\_languages | status | tagline | title\_x | vote\_average | vote\_count | movie\_id | title\_y | cast | crew |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2683 | 14000000 | [{"id": 28, "name": "Action"}, {"id": 12, "nam... | NaN | 13950 | [{"id": 633, "name": "record store"}, {"id": 7... | en | All About the Benjamins | Bucum Jackson (Cube) is a bounty hunter with a... | 5.102258 | [{"name": "New Line Cinema", "id": 12}] | ... | [{"iso\_639\_1": "en", "name": "English"}, {"iso... | Released | Money can make people do funny things. | All About the Benjamins | 5.8 | 48 | 13950 | All About the Benjamins | [{"cast\_id": 1, "character": "Bucum", "credit\_... | [{"credit\_id": "52fe45b99251416c7506049d", "de... |
| 521 | 60000000 | [{"id": 35, "name": "Comedy"}, {"id": 18, "nam... | http://www.theterminal-themovie.com/ | 594 | [{"id": 242, "name": "new york"}, {"id": 822, ... | en | The Terminal | Viktor Navorski is a man without a country; hi... | 57.753914 | [{"name": "DreamWorks SKG", "id": 27}, {"name"... | ... | [{"iso\_639\_1": "bg", "name": "\u0431\u044a\u04... | Released | Life is waiting. | The Terminal | 7.0 | 1910 | 594 | The Terminal | [{"cast\_id": 4, "character": "Viktor Navorski"... | [{"credit\_id": "52fe4259c3a36847f8017699", "de... |
| 167 | 130000000 | [{"id": 28, "name": "Action"}, {"id": 12, "nam... | NaN | 7364 | [{"id": 168297, "name": "tyrant"}, {"id": 1683... | en | Sahara | Scouring the ocean depths for treasure-laden s... | 21.605568 | [{"name": "Paramount Pictures", "id": 4}, {"na... | ... | [{"iso\_639\_1": "en", "name": "English"}, {"iso... | Released | Dirk Pitt. Adventure has a new name. | Sahara | 5.7 | 434 | 7364 | Sahara | [{"cast\_id": 1, "character": "Dirk Pitt", "cre... | [{"credit\_id": "536b5cb40e0a2647db00b826", "de... |
| 3111 | 0 | [{"id": 35, "name": "Comedy"}, {"id": 80, "nam... | NaN | 11546 | [{"id": 282, "name": "video game"}, {"id": 103... | en | Police Academy: Mission to Moscow | The Russians need help in dealing with the Maf... | 11.946898 | [{"name": "Warner Bros.", "id": 6194}] | ... | [{"iso\_639\_1": "en", "name": "English"}] | Released | Just when we thought the Cold War was over, le... | Police Academy: Mission to Moscow | 4.1 | 178 | 11546 | Police Academy: Mission to Moscow | [{"cast\_id": 9, "character": "Eric Lassard", "... | [{"credit\_id": "52fe44569251416c750313e3", "de... |
| 901 | 48000000 | [{"id": 12, "name": "Adventure"}] | NaN | 8367 | [{"id": 392, "name": "england"}, {"id": 2868, ... | en | Robin Hood: Prince of Thieves | When the dastardly Sheriff of Nottingham murde... | 28.803729 | [{"name": "Warner Bros.", "id": 6194}, {"name"... | ... | [{"iso\_639\_1": "en", "name": "English"}] | Released | For the good of all men, and the love of one w... | Robin Hood: Prince of Thieves | 6.6 | 909 | 8367 | Robin Hood: Prince of Thieves | [{"cast\_id": 1, "character": "Robin Hood", "cr... | [{"credit\_id": "52fe44a3c3a36847f80a17fb", "de... |

5 rows × 24 columns

def to\_list(df, feature\_names\_list): *#df: dataframe, feature\_names: list of all features to convert from JSON to list*

for feature\_name **in** feature\_names\_list:

print("Current:", feature\_name)

*#STEP 1: convert JSON format to a list*

df[feature\_name]=df[feature\_name].apply(json.loads)

*#Two cases here: Feature is crew, or feature is not crew*

if feature\_name == 'crew': *#if crew, due to large size, want to limit to most influential members: director, editor, cinematographer, screenplay, and composer*

for index,i **in** zip(df.index,df[feature\_name]):

feature\_list\_1=[]

limit = 10

if len(i) < 10:

limit = len(i)

for j **in** range(limit): *#top 10 crew members*

feature\_list\_1.append((i[j]['name'])) *# the key 'name' contains the name of the a sub-feature (ex: sci-fi in genres)*

df.loc[index,feature\_name]= str(feature\_list\_1)

elif feature\_name == 'cast': *#Another special case. Only want top 5 cast members (most infulential)*

for index,i **in** zip(df.index,df[feature\_name]):

feature\_list\_1=[]

if len(i) > 5:

limit = 5

else:

limit = len(i)

for j **in** range(limit): *#top 5 (JSON format already has this sorted)*

feature\_list\_1.append((i[j]['name']))

df.loc[index,feature\_name]= str(feature\_list\_1)

else:

for index,i **in** zip(df.index,df[feature\_name]):

feature\_list\_1=[]

for j **in** range(len(i)):

feature\_list\_1.append((i[j]['name']))

df.loc[index,feature\_name]= str(feature\_list\_1)

*#STEP 2: clean up and transform into unsorted list*

df[feature\_name] = df[feature\_name].str.strip('[]').str.replace(' ','').str.replace("'",'')

df[feature\_name] = df[feature\_name].str.split(',')

*#STEP 3: Sort list elements*

for i,j **in** zip(df[feature\_name],df.index):

features\_list\_2=i

features\_list\_2.sort()

df.loc[j,feature\_name]=str(features\_list\_2)

df[feature\_name]=df[feature\_name].str.strip('[]').str.replace(' ','').str.replace("'",'')

lst = df[feature\_name].str.split(',')

if len(lst) == 0:

df[feature\_name] = None

else:

df[feature\_name]= df[feature\_name].str.split(',')

return df

movies = to\_list(movies, ['genres', 'keywords', 'production\_companies', 'cast', 'crew'])

Current: genres

Current: keywords

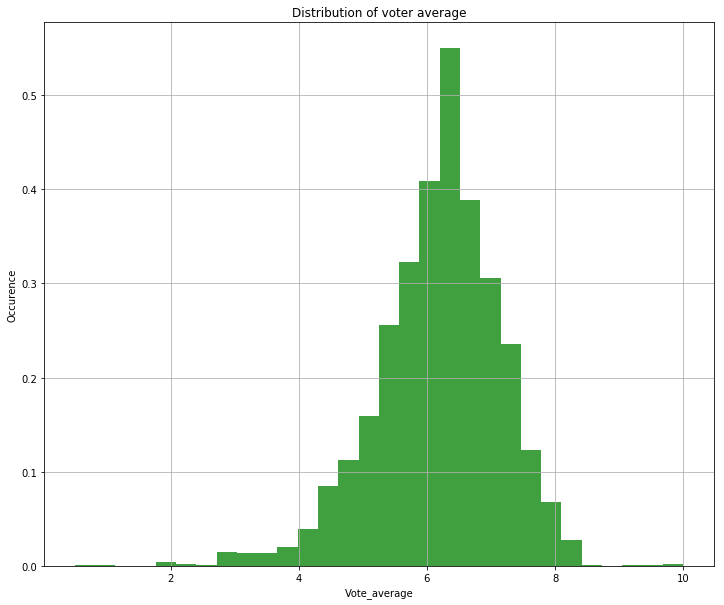
Current: production\_companies

Current: cast

Current: crew

plt.subplots(figsize=(12,10))n, bins, patches = plt.hist(movies\_shortened['vote\_average'], 30, density=1, facecolor='g', alpha=0.75)

plt.xlabel('Vote\_average')plt.ylabel('Occurence')plt.title('Distribution of voter average')plt.grid(True)plt.show()print("Minimum of Ratings:", round(min(movies\_shortened['vote\_average']),2))print("Maximum of Ratings:", round(max(movies\_shortened['vote\_average']),2))print("Average of Ratings:", round(np.mean(movies\_shortened['vote\_average']),2))print("Variance of Ratings:",round(np.var(movies\_shortened['vote\_average']),2))



Minimum of Ratings: 0.5

Maximum of Ratings: 10.0

Average of Ratings: 6.17

Variance of Ratings: 0.93

fig, ax = plt.subplots(2,2, figsize=(24,20))

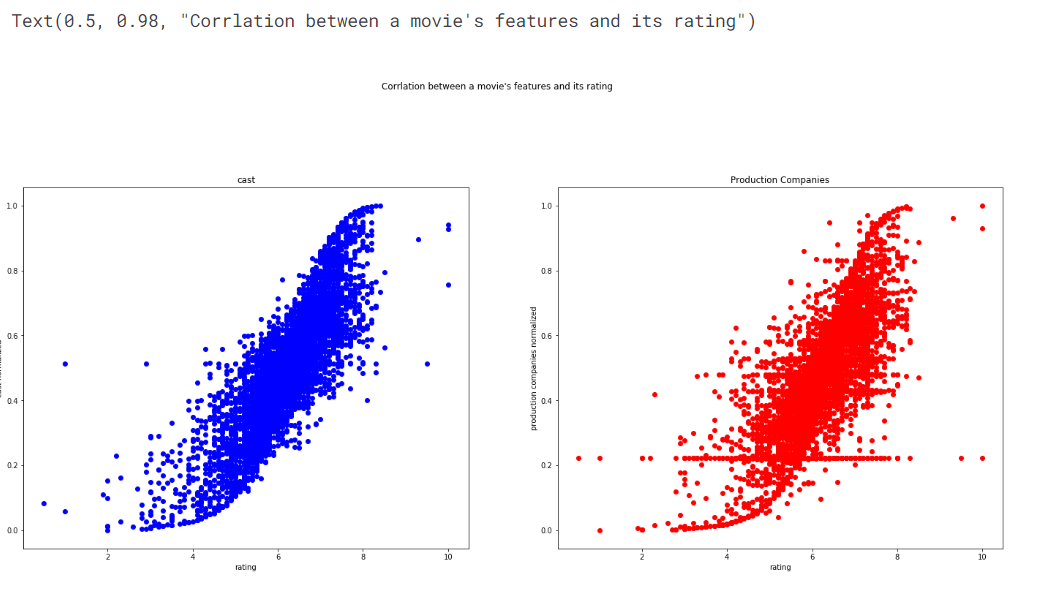
ax[0,0].scatter(target\_df['ratings'], feat\_scaled['cast'], facecolor='blue')ax[0,0].set\_xlabel('rating')ax[0,0].set\_ylabel('cast normalized')ax[0,0].set\_title('cast')

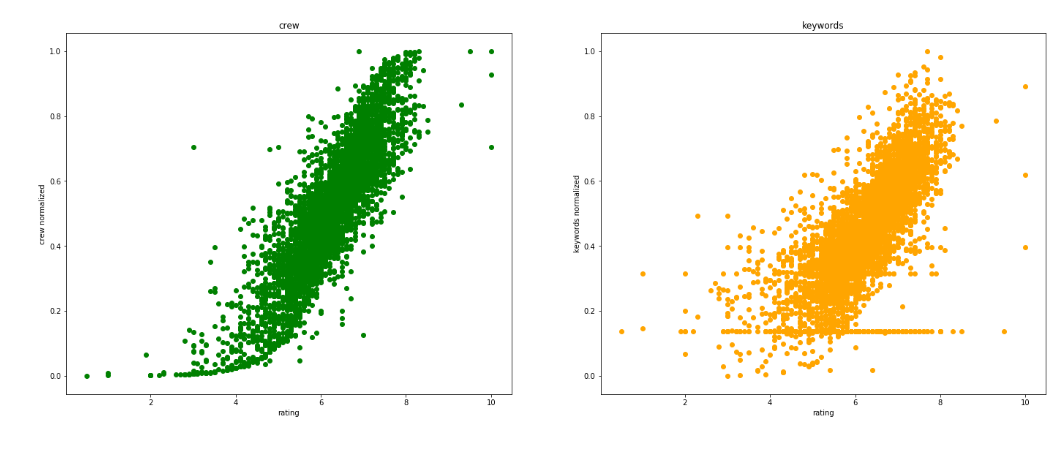
ax[1,0].scatter(target\_df['ratings'], feat\_scaled['crew'], facecolor='green')ax[1,0].set\_xlabel('rating')ax[1,0].set\_ylabel('crew normalized')ax[1,0].set\_title('crew')

ax[0,1].scatter(target\_df['ratings'], feat\_scaled['production\_companies'], facecolor='red')ax[0,1].set\_xlabel('rating')ax[0,1].set\_ylabel('production companies normalized')ax[0,1].set\_title('Production Companies')

ax[1,1].scatter(target\_df['ratings'], feat\_scaled['keywords'], facecolor='orange')ax[1,1].set\_xlabel('rating')ax[1,1].set\_ylabel('keywords normalized')ax[1,1].set\_title('keywords')

fig.suptitle("Corrlation between a movie's features and its rating")





from sklearn.metrics import r2\_score

score = r2\_score(target\_test, target\_pred)

print("R^2 Score for predictions:", score)

R^2 Score for predictions: 0.7976039015153281

Task 3

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

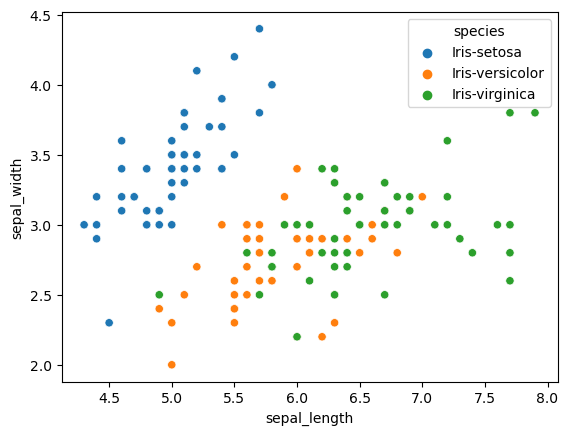
import warningswarnings.filterwarnings('ignore')

iris = pd.read\_csv(r'/kaggle/input/iris-flower-dataset/IRIS.csv')

Iris

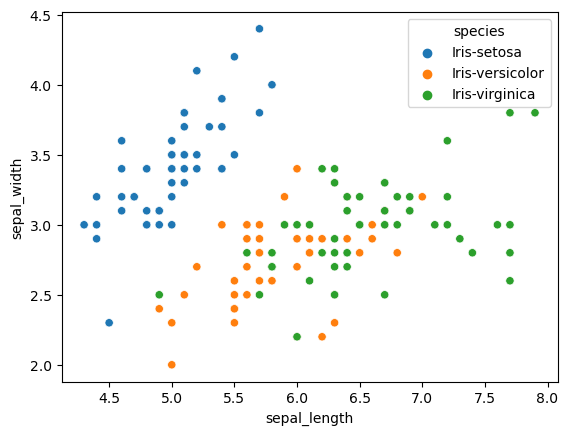
| sepal\_length | sepal\_width | petal\_length | petal\_width | species |
| --- | --- | --- | --- | --- |
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| ... | ... | ... | ... | ... | ... |
| 145 | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica |
| 146 | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica |
| 147 | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica |
| 148 | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica |
| 149 | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica |
|  |  |  |  |  |  |

sns.scatterplot(x='sepal\_length', y='sepal\_width', hue='species', data=iris)plt.show()

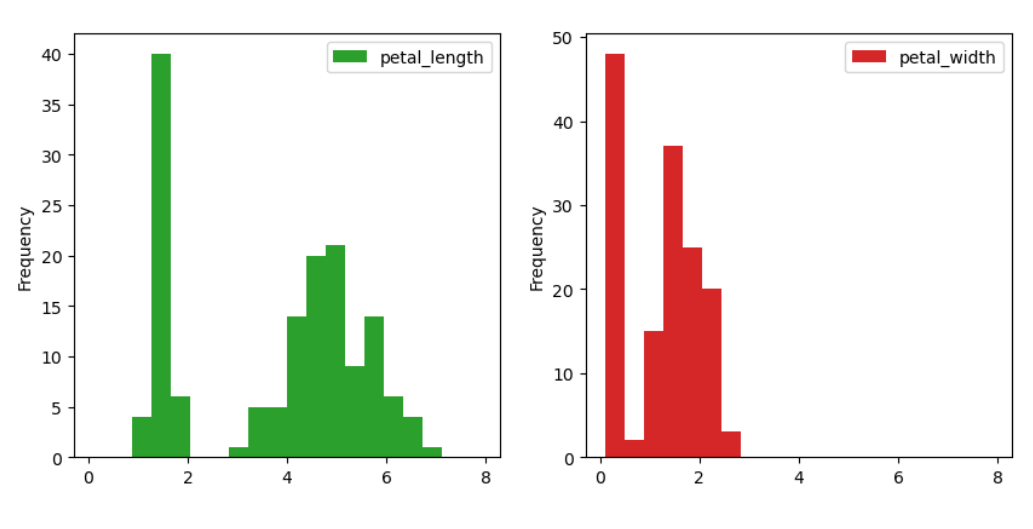


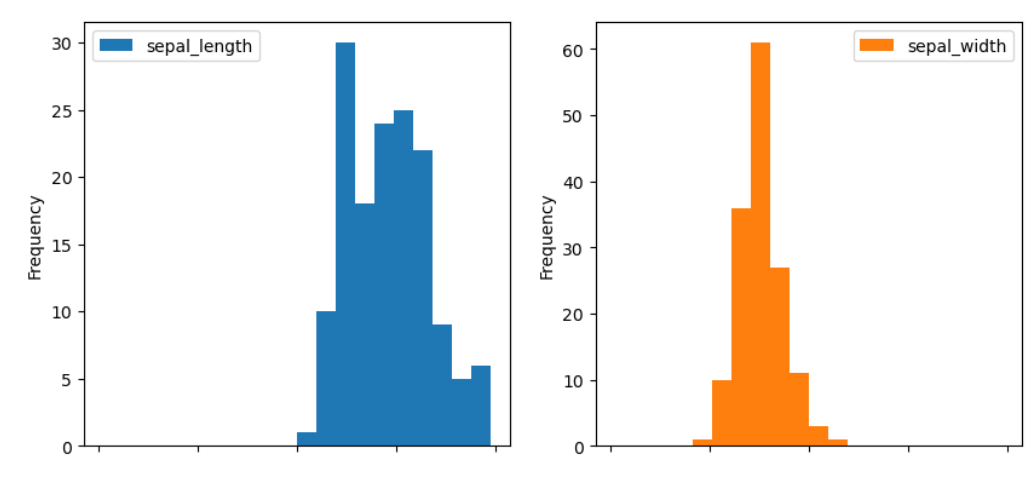
sns.lineplot(data=iris.drop(['species'], axis=1))

plt.show()

iris.plot.hist(subplots=True, layout=(2,2), figsize=(10, 10), bins=20)

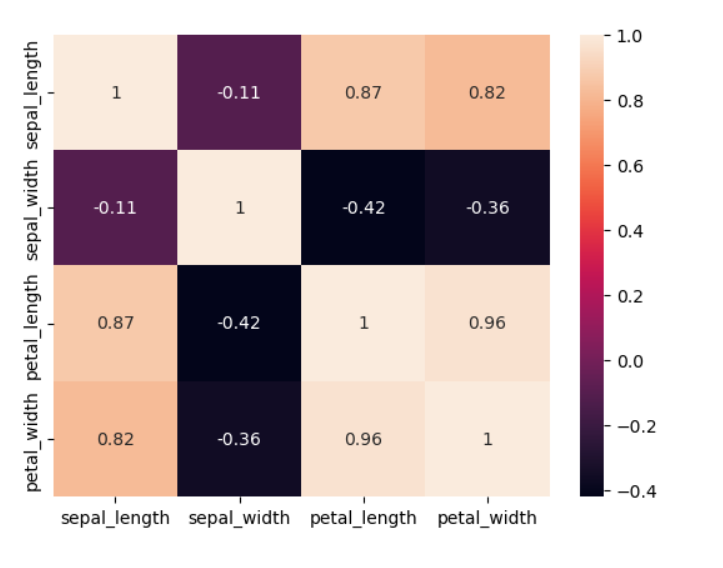
plt.show()





sns.heatmap(iris.corr(), annot=True

plt.show()



from sklearn.tree import DecisionTreeClassifierdtree = DecisionTreeClassifier()dtree.fit(x\_train, y\_train)

dtree.score(x\_test, y\_test)

Out[20]:

0.95

Task 4

import numpy as np

*# linear algebra*import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

import tensorflow as tf

import matplotlib.pyplot as pltimport seaborn as snsfrom sklearn.manifold import TSNEfrom sklearn.decomposition import PCA, TruncatedSVDimport matplotlib.patches as mpatchesimport time

*# Classifier Libraries*from sklearn.linear\_model import LogisticRegressionfrom sklearn.svm import SVCfrom sklearn.neighbors import KNeighborsClassifierfrom sklearn.tree import DecisionTreeClassifierfrom sklearn.ensemble import RandomForestClassifierimport collections

*# Other Libraries*from sklearn.model\_selection import train\_test\_splitfrom sklearn.pipeline import make\_pipelinefrom imblearn.pipeline import make\_pipeline as imbalanced\_make\_pipelinefrom imblearn.over\_sampling import SMOTEfrom imblearn.under\_sampling import NearMissfrom imblearn.metrics import classification\_report\_imbalancedfrom sklearn.metrics import precision\_score, recall\_score, f1\_score, roc\_auc\_score, accuracy\_score, classification\_reportfrom collections import Counterfrom sklearn.model\_selection import KFold, StratifiedKFoldimport warningswarnings.filterwarnings("ignore")

df = pd.read\_csv('../input/creditcard.csv')df.head()

| Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 | V11 | V12 | V13 | V14 | V15 | V16 | V17 | V18 | V19 | V20 | V21 | V22 | V23 | V24 | V25 | V26 | V27 | V28 | Amount | Class |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0.0 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 | 0.239599 | 0.098698 | 0.363787 | 0.090794 | -0.551600 | -0.617801 | -0.991390 | -0.311169 | 1.468177 | -0.470401 | 0.207971 | 0.025791 | 0.403993 | 0.251412 | -0.018307 | 0.277838 | -0.110474 | 0.066928 | 0.128539 | -0.189115 | 0.133558 | -0.021053 | 149.62 | 0 |
| 1 | 0.0 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.078803 | 0.085102 | -0.255425 | -0.166974 | 1.612727 | 1.065235 | 0.489095 | -0.143772 | 0.635558 | 0.463917 | -0.114805 | -0.183361 | -0.145783 | -0.069083 | -0.225775 | -0.638672 | 0.101288 | -0.339846 | 0.167170 | 0.125895 | -0.008983 | 0.014724 | 2.69 | 0 |
| 2 | 1.0 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.791461 | 0.247676 | -1.514654 | 0.207643 | 0.624501 | 0.066084 | 0.717293 | -0.165946 | 2.345865 | -2.890083 | 1.109969 | -0.121359 | -2.261857 | 0.524980 | 0.247998 | 0.771679 | 0.909412 | -0.689281 | -0.327642 | -0.139097 | -0.055353 | -0.059752 | 378.66 | 0 |
| 3 | 1.0 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 | 0.237609 | 0.377436 | -1.387024 | -0.054952 | -0.226487 | 0.178228 | 0.507757 | -0.287924 | -0.631418 | -1.059647 | -0.684093 | 1.965775 | -1.232622 | -0.208038 | -0.108300 | 0.005274 | -0.190321 | -1.175575 | 0.647376 | -0.221929 | 0.062723 | 0.061458 | 123.50 | 0 |
| 4 | 2.0 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.592941 | -0.270533 | 0.817739 | 0.753074 | -0.822843 | 0.538196 | 1.345852 | -1.119670 | 0.175121 | -0.451449 | -0.237033 | -0.038195 | 0.803487 | 0.408542 | -0.009431 | 0.798278 | -0.137458 | 0.141267 | -0.206010 | 0.502292 | 0.219422 | 0.215153 | 69.99 | 0 |

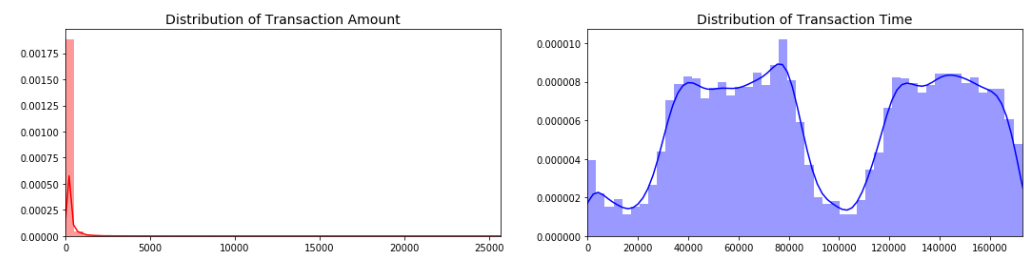
fig, ax = plt.subplots(1, 2, figsize=(18,4))

amount\_val = df['Amount'].valuestime\_val = df['Time'].values

sns.distplot(amount\_val, ax=ax[0], color='r')ax[0].set\_title('Distribution of Transaction Amount', fontsize=14)ax[0].set\_xlim([min(amount\_val), max(amount\_val)])

sns.distplot(time\_val, ax=ax[1], color='b')ax[1].set\_title('Distribution of Transaction Time', fontsize=14)ax[1].set\_xlim([min(time\_val), max(time\_val)])

plt.show()



print('Distribution of the Classes in the subsample dataset')print(new\_df['Class'].value\_counts()/len(new\_df))

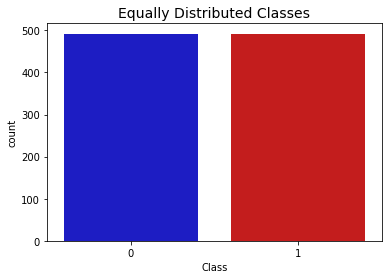
sns.countplot('Class', data=new\_df, palette=colors)plt.title('Equally Distributed Classes', fontsize=14)plt.show()

Distribution of the Classes in the subsample dataset

1 0.5

0 0.5

Name: Class, dtype: float64



from scipy.stats import norm

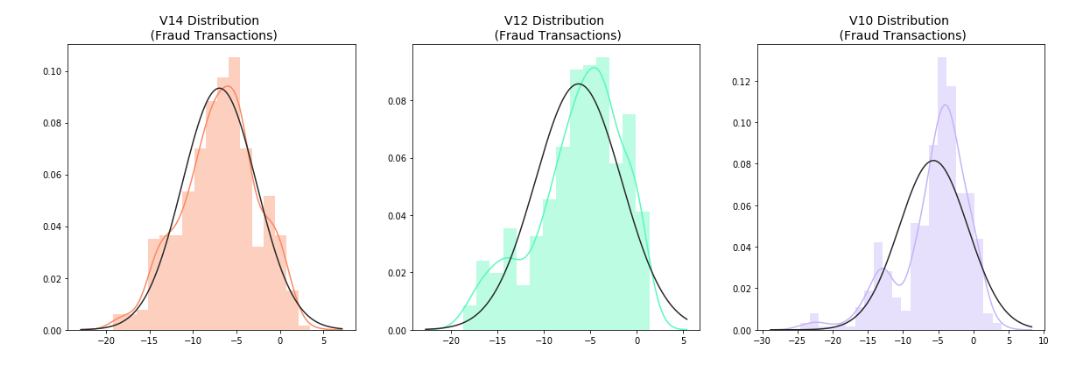
f, (ax1, ax2, ax3) = plt.subplots(1,3, figsize=(20, 6))

v14\_fraud\_dist = new\_df['V14'].loc[new\_df['Class'] == 1].valuessns.distplot(v14\_fraud\_dist,ax=ax1, fit=norm, color='#FB8861')ax1.set\_title('V14 Distribution **\n** (Fraud Transactions)', fontsize=14)

v12\_fraud\_dist = new\_df['V12'].loc[new\_df['Class'] == 1].valuessns.distplot(v12\_fraud\_dist,ax=ax2, fit=norm, color='#56F9BB')ax2.set\_title('V12 Distribution **\n** (Fraud Transactions)', fontsize=14)

v10\_fraud\_dist = new\_df['V10'].loc[new\_df['Class'] == 1].valuessns.distplot(v10\_fraud\_dist,ax=ax3, fit=norm, color='#C5B3F9')ax3.set\_title('V10 Distribution **\n** (Fraud Transactions)', fontsize=14)

plt.show()



f, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(24,6))*# labels = ['No Fraud', 'Fraud']*f.suptitle('Clusters using Dimensionality Reduction', fontsize=14)

blue\_patch = mpatches.Patch(color='#0A0AFF', label='No Fraud')red\_patch = mpatches.Patch(color='#AF0000', label='Fraud')

*# t-SNE scatter plot*ax1.scatter(X\_reduced\_tsne[:,0], X\_reduced\_tsne[:,1], c=(y == 0), cmap='coolwarm', label='No Fraud', linewidths=2)ax1.scatter(X\_reduced\_tsne[:,0], X\_reduced\_tsne[:,1], c=(y == 1), cmap='coolwarm', label='Fraud', linewidths=2)ax1.set\_title('t-SNE', fontsize=14)

ax1.grid(True)

ax1.legend(handles=[blue\_patch, red\_patch])

*# PCA scatter plot*ax2.scatter(X\_reduced\_pca[:,0], X\_reduced\_pca[:,1], c=(y == 0), cmap='coolwarm', label='No Fraud', linewidths=2)ax2.scatter(X\_reduced\_pca[:,0], X\_reduced\_pca[:,1], c=(y == 1), cmap='coolwarm', label='Fraud', linewidths=2)ax2.set\_title('PCA', fontsize=14)

ax2.grid(True)

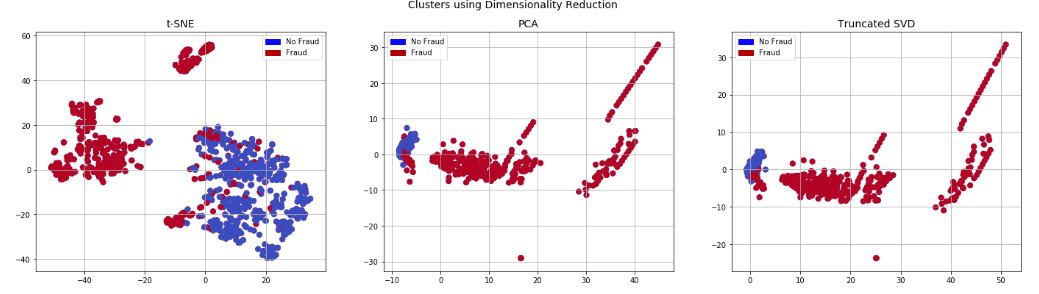
ax2.legend(handles=[blue\_patch, red\_patch])

*# TruncatedSVD scatter plot*ax3.scatter(X\_reduced\_svd[:,0], X\_reduced\_svd[:,1], c=(y == 0), cmap='coolwarm', label='No Fraud', linewidths=2)ax3.scatter(X\_reduced\_svd[:,0], X\_reduced\_svd[:,1], c=(y == 1), cmap='coolwarm', label='Fraud', linewidths=2)ax3.set\_title('Truncated SVD', fontsize=14)

ax3.grid(True)

ax3.legend(handles=[blue\_patch, red\_patch])

plt.show()



from sklearn.metrics import confusion\_matrix

*# Logistic Regression fitted using SMOTE technique*y\_pred\_log\_reg = log\_reg\_sm.predict(X\_test)

*# Other models fitted with UnderSampling*y\_pred\_knear = knears\_neighbors.predict(X\_test)y\_pred\_svc = svc.predict(X\_test)y\_pred\_tree = tree\_clf.predict(X\_test)

log\_reg\_cf = confusion\_matrix(y\_test, y\_pred\_log\_reg)kneighbors\_cf = confusion\_matrix(y\_test, y\_pred\_knear)svc\_cf = confusion\_matrix(y\_test, y\_pred\_svc)tree\_cf = confusion\_matrix(y\_test, y\_pred\_tree)

fig, ax = plt.subplots(2, 2,figsize=(22,12))

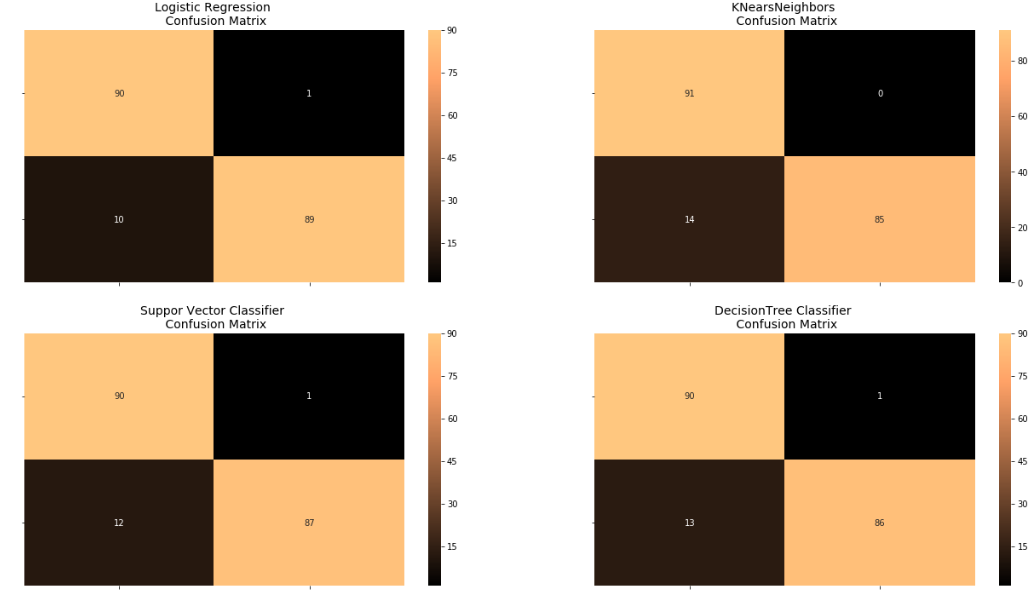
sns.heatmap(log\_reg\_cf, ax=ax[0][0], annot=True, cmap=plt.cm.copper)ax[0, 0].set\_title("Logistic Regression **\n** Confusion Matrix", fontsize=14)ax[0, 0].set\_xticklabels(['', ''], fontsize=14, rotation=90)ax[0, 0].set\_yticklabels(['', ''], fontsize=14, rotation=360)

sns.heatmap(kneighbors\_cf, ax=ax[0][1], annot=True, cmap=plt.cm.copper)ax[0][1].set\_title("KNearsNeighbors **\n** Confusion Matrix", fontsize=14)ax[0][1].set\_xticklabels(['', ''], fontsize=14, rotation=90)ax[0][1].set\_yticklabels(['', ''], fontsize=14, rotation=360)

sns.heatmap(svc\_cf, ax=ax[1][0], annot=True, cmap=plt.cm.copper)ax[1][0].set\_title("Suppor Vector Classifier **\n** Confusion Matrix", fontsize=14)ax[1][0].set\_xticklabels(['', ''], fontsize=14, rotation=90)ax[1][0].set\_yticklabels(['', ''], fontsize=14, rotation=360)

sns.heatmap(tree\_cf, ax=ax[1][1], annot=True, cmap=plt.cm.copper)ax[1][1].set\_title("DecisionTree Classifier **\n** Confusion Matrix", fontsize=14)ax[1][1].set\_xticklabels(['', ''], fontsize=14, rotation=90)ax[1][1].set\_yticklabels(['', ''], fontsize=14, rotation=360)

plt.show()



oversample\_predictions = oversample\_model.predict(original\_Xtest, batch\_size=200, verbose=0)

oversample\_fraud\_predictions = oversample\_model.predict\_classes(original\_Xtest, batch\_size=200, verbose=0)

oversample\_smote = confusion\_matrix(original\_ytest, oversample\_fraud\_predictions)actual\_cm = confusion\_matrix(original\_ytest, original\_ytest)labels = ['No Fraud', 'Fraud']

fig = plt.figure(figsize=(16,8))

fig.add\_subplot(221)plot\_confusion\_matrix(oversample\_smote, labels, title="OverSample (SMOTE) **\n** Confusion Matrix", cmap=plt.cm.Oranges)

fig.add\_subplot(222)plot\_confusion\_matrix(actual\_cm, labels, title="Confusion Matrix **\n** (with 100**% a**ccuracy)", cmap=plt.cm.Greens)

Confusion matrix, without normalization

[[56851 12]

[ 33 65]]

Confusion matrix, without normalization

[[56863 0]

[ 0 98]]

