INSURANCE FRAUD DETECTION

#Data Collection

```
    we have insurance claims data (insurance claim.csv). However, this data is not perfect.
```

· It may contain outliers, missing values and irrelevant (not related to insurance claims) information

```
In [1]: import warnings
                            warnings.filterwarnings(action='ignore', category=FutureWarning)
In [2]: # importing required libraries
                           import pandas as pd
                           import numpy as np
                           import matplotlib.pvplot as plt
                           import seaborn as sns
                           # To get all the rows and columns
                          pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
In [3]: # Reading the Dataset File
                           df=pd.read_csv('insurance_claims.csv' )
In [4]: df.head()
Out[4]:
                                      months_as_customer age policy_number policy_bind_date policy_state policy_csl policy_deductable policy_annumber policy_bind_date policy_state policy_csl policy_deductable policy_annumber policy_bind_date policy_state policy_csl policy_deductable policy_state policy_state policy_csl policy_deductable policy_state policy_state policy_csl policy_deductable policy_state 
                             0
                                                                                 328
                                                                                                                                521585
                                                                                                                                                                     2014-10-17
                                                                                                                                                                                                                          ОН
                                                                                                                                                                                                                                            250/500
                                                                                                                                                                                                                                                                                                     1000
                                                                                 228
                                                                                                42
                                                                                                                                342868
                                                                                                                                                                    2006-06-27
                                                                                                                                                                                                                           IN
                                                                                                                                                                                                                                            250/500
                                                                                                                                                                                                                                                                                                     2000
                                                                                                                                                                    2000-09-06
                                                                                   134
                                                                                                                                687698
                                                                                                                                                                                                                          ОН
                                                                                                                                                                                                                                            100/300
                                                                                                                                                                                                                                                                                                     2000
                                                                                 256
                                                                                                                                227811
                                                                                                                                                                     1990-05-25
                                                                                                                                                                                                                            IL
                                                                                                                                                                                                                                            250/500
                                                                                                                                                                                                                                                                                                     2000
                                                                                                                                367455
                                                                                 228
                                                                                                                                                                    2014-06-06
                                                                                                                                                                                                                             IL 500/1000
                                                                                                                                                                                                                                                                                                     1000
In [5]: # Check the columns
                                     _c39 is extra columns let's remove it
                           df.drop('_c39', axis = 1, inplace=True)
In [6]: df.head()
Out[6]:
                                      months_as_customer age policy_number policy_bind_date policy_state policy_csl policy_deductable policy_annumber
                                                                                 328
                                                                                                                                342868
                                                                                                                                                                     2006-06-27
                                                                                                                                                                                                                                            250/500
                                                                                                                                                                                                                                                                                                     2000
                                                                                                                                687698
                                                                                                                                                                     2000-09-06
                                                                                                                                                                                                                                            100/300
                                                                                                                                                                                                                                                                                                     2000
                                                                                                                                227811
                                                                                                                                                                      1990-05-25
                                                                                                                                                                                                                                            250/500
                                                                                                                                                                                                                                                                                                     2000
                                                                                 228
                                                                                                                                367455
                                                                                                                                                                     2014-06-06
                                                                                                                                                                                                                                         500/1000
                                                                                                                                                                                                                                                                                                     1000
IN [3]: defadsofibsfbiteninformation(df, column_name):
    df.describe()
                                                     Function is responsible for printing out basic information
Out[7]:
                                                about the dataset such as Null values, Data types, etc. months_as_customer age policy_number policy_deductable policy_annual_premium umbrella_limit
                              countata_type 1000ff[00000mm1_000.00000type1000.000000
                                                                                                                                                                                                   1000.000000
                                                                                                                                                                                                                                                                   1000.000000 1.000000e+03
                               countata_type rousencoses in a second in a
                                                                                                                                                                                                  1136.000000
                                                                                                                                                                                                                                                                   1256.406150 1.101000e+06 5
                                                                                                                                                                                                     611.864673
                                                                                                                                                                                                                                                                   244.167395 2.297407e+06
                                   print("############################"\n")
miprint("column 99990", 291999991am19804.00000
                                                                                                                                                                                                      500.000000
                                                                                                                                                                                                                                                                    433.330000 -1.000000e+06 43
                                 25%rint("data<sub>1</sub>ts/%56000 data_05000000 500.000000 print("total number of null values:", null values)
50%print() 199.500000 38.000000 533135.000000 1000.000000
                                                                                                                                                                                                                                                                   1089.607500 0.000000e+00 44
                                                                                                                                                                                                                                                                   1257.200000 0.000000e+00 46
                           75% 276.250000 44.000000 759099.750000 column_headers = list(df.columns) fomax in column170e00000 64.000000 999435.000000
                                                                                                                                                                                                    2000.000000
                                                                                                                                                                                                                                                                   1415.695000 0.000000e+00 60
                                                                                                                                                                                                    2000.000000
                                                                                                                                                                                                                                                                   2047.590000 1.000000e+07 62
                             display_basic_information(df,i)
In [8]:
                           ##\\#mban#afi#กลพระสกุป columns of data
                           Baite("INTOPMAteforews:", len(df))
printfillinghammafinaplumns:", len(df.columns))
                           ฟิษาปัตก คร์meowmontAgoas_customer
ฟิษาปัตว เรอินบุตร: 39
                            total number of null values: 0
                           Data Preparation
                           ############################

• We prepare the data column wise, as every column would require different preprocessing
                           column name: age
                           total number of null values: 0
```

```
Function is responsible for printing out basic information
    Out[7]:
                                                   about the dataset such as Null values, Data types, etc.
""months_as_customer age policy_number policy_deductable policy_annual_premium umbrella_limit
                                   countata_type +0.0df0[000domn1_000med]oottype1000.000000
                                                                                                                                                                                                          1000.000000
                                                                                                                                                                                                                                                                         1000.000000 1.000000e+03
                                    1136.000000
                                                                                                                                                                                                                                                                         1256.406150 1.101000e+06 5
                                     611.864673
                                                                                                                                                                                                                                                                         244.167395 2.297407e+06
                                                                                                                                                                                                            500.000000
                                                                                                                                                                                                                                                                         433.330000 -1.000000e+06 43
                                                                                                                                                                                                            500.000000
                                                                                                                                                                                                                                                                        1089.607500 0.000000e+00 44
                                      50%rint()
                                                                                                                                                                                                          1000.000000
                                                                                                                                                                                                                                                                       1257.200000 0.000000e+00 46
                                75% 276.250000 44.00000 759099.750000 column_headers = list(df.columns) for the column column ( \frac{7}{16} \frac{1}{16} \frac
                                                                                                                                                                                                         2000.000000
                                                                                                                                                                                                                                                                        1415.695000 0.000000e+00 60
                                                                                                                                                                                                         2000.000000
                                                                                                                                                                                                                                                                       2047.590000 1.000000e+07 62
                                 display_basic_information(df,i)
    ⊎ยบบลก คล์mย®พลonใA90as_customer
ฟยซอะซyดิธ์:cฐฟิชุธร: 39
                                total number of null values: 0
                                 Data Preparation
                                **#######################

• We prepare the data column wise, as every column would require different preprocessing
                                 column name: age
                                 data type: int64
                                total number of null values: 0
                                *****************
                                Basic Information:
   In [10]: # To check the number of unique values inside the dataframe
                                 for col in df.columns:
                                            print(col, df[col].dtype , df[col].nunique())
                                months_as_customer int64 391
                                 age int64 46
                                policy number int64 1000
                                policy_bind_date object 951
                                policy_state object 3
policy_csl object 3
                                policy_deductable int64 3
policy_annual_premium float64 991
                                umbrella_limit int64 11 insured_zip int64 995
                                 insured_sex object 2
                                 insured_education_level object 7
                                 insured_occupation object 14
                                 insured hobbies object 20
                                 insured_relationship object 6
                                capital-gains int64 338 capital-loss int64 354
                                incident_date object 60
incident_type object 4
                                collision_type object 4
incident_severity object 4
                                 authorities_contacted object 5
                                 incident state object 7
                                 incident_city object 7
                                 incident location object 1000
                                 incident_hour_of_the_day int64 24
                                number of vehicles involved int64 4
                                property_damage object 3
bodily_injuries int64 3
                                witnesses int64 4

Checking the iquality of data
                                 total_claim_amount int64 763
                                injury claim int64 638
# Checking the null 4 02 gree
property claim int64 626
Venicle 2 134m (int64 726
                                auto make object 14
months as customer
auto model object 39
  Out[13]:
                                                                                                                                    а
                                alto year int64 21
policy state
fraud reported object 2
policy_deductable
                                                                                                                                    0
                                policy_annual_premium
   In [11]:
                                #mbrellahdimbeve we can see that policy_number, policy_csl, policy_bind_date,
                                 #nsaredeziplocation, insured_hobbies, incident_date, incident_city, auto_model
                                 \begin{array}{ll} \hbox{\tt Ensured\_bex} helpful \ in \ knowledge@extraction \ hence \ removing \ those.} \\ \hbox{\tt insured\_education\_level} & 0 \end{array} 
                               incident type the drop method is successful the trype the trype the trype trype the trype 
   In [12]:
GhtshempSeverity ወ
authorities contacted
Index66tmgqfte_as_customer', 'age', 'policy_state', 'policy_deductable',
incident PAδifyoantHelaggemium', 'umbrella_limit', 'insured_zip', 'insured_sex',
number_ofnSthfteleduffitonedevelj, 'insured_occupation', 'insured_relationship',
property'aBhigl-gains', 'capitaj-loss', 'incident_type', 'collision_type',
bodily_inflintest_severity', 'authorities_contacted', 'incident_state',
witnessesincident_hour_of_the_day', 'number_of_vehicles_involved',
police_reporersyaftmomes', 'bodidy_injuries', 'witnesses',
total_cignlinenty_claim', 'vehicle_claim', 'witnesses',
total_cignlinenty_claim', 'vehicle_claim', 'auto_make', 'auto_year',
property'felysim'eported'], 0
vehicletysim'eported'), 0
auto make 0
                                 auto make
```

```
bodily_injuries int64 3
                                                              witnesses int64 4
Checking the iquality cof data
                                                              total_claim_amount int64 763
injury_claim_int64 638
# Checkingline nutl values
property_claim_int64 626
Venicle_claim_int64 726
     In [13]:
                                                           auto make object 14
months accustomer
auto model object 39
age
auto model object 39
age
auto model object 21
policy sported object 2
policy_deductable
     Out[13]:
                                                                                                                                                                                                                                                                 0
                                                                                                                                                                                                                                                                 0
                                                                policy_annual_premium
  In [11]: #mbrealthtimiteve we can see that policy_number, policy_csl, policy_bind_date, #nsuredeziplocation, insured_hobbies, incident_date, incident_city, auto_model
                                                                ₫nsarėd_bexhelpful in knowledge⊙extraction hence removing those
                                                                insured education level
                                                              dfsdred_Geropaty_number', 'polioy_csl', 'policy_bind_date', 'incident_location',\
insured_relationshipobies', 'inpident_date', 'incident_city', 'auto_model'],\
capital-gains1, inplace=True) 0
                                                                capital-loss
                                                             incident type the drop method is succesful the successful the succ
    In [12]:
dfcfdehtnseverity
authorities contacted
Individate contacted
Individate
Individate contacted
Individate
In
                                                              auto make
                                                                auto_year
                                                                 fraud reported
                                                                                                                                                                                                                                                                 0
                                                                dtype: int64
    In [14]: # Check the values present inside the object type columns
                                                              for col in df.columns:
                                                                                       if df[col].dtype == 'object':
                                                                                                                print(col, df[col].unique())
                                                             policy_state ['OH' 'IN' 'IL']
insured_sex ['MALE' 'FEMALE']
insured_ducation_level ['MD' 'PhD' 'Associate' 'Masters' 'High School' 'College' 'JD']
insured_occupation ['craft-repair' 'machine-op-inspct' 'sales' 'armed-forces' 'tech-supp'
'prof-specialty' 'other-service' 'priv-house-serv' 'exec-managerial'
'protective-serv' 'transport-moving' 'handlers-cleaners' 'adm-clerical'
                                                                         'farming-fishing']
                                                              insured relationship ['husband' 'other-relative' 'own-child' 'unmarried' 'wife' 'not-in-famil
                                                                incident_type ['Single Vehicle Collision' 'Vehicle Theft' 'Multi-vehicle Collision'
                                                                collision_type ['Side Collision' '?' 'Rear Collision' 'Front Collision']
                                                           collision_type ['Side Collision' '?' 'Rear Collision' 'Front Collision']
incident_severity ['Major Damage' 'Minor Damage' 'Total Loss' 'Trivial Damage']
authorities_contacted ['Police' 'None' 'Fire' 'Other' 'Ambulance']
incident_state ['SC' 'VA' 'NY' 'OH' 'WV' 'NC' 'PA']
property_damage ['YES' '?' 'NO']
police_report_available ['YES' '?' 'NO']
auto_make ['Saab' 'Mercedes' 'Dodge' 'Chevrolet' 'Accura' 'Nissan' 'Audi' 'Toyota'
'Ford' 'Suburu' 'BMW' 'Jeep' 'Honda' 'Volkswagen']
fraud_reported ['Y' 'N']
    In [15]: # collision_type, property_damage, police_report_available has '?' symbol.
# We can treat the '?' as new category with 'Unknown' value.
                                                           df['collision_type'].replace('?','Not Known', inplace=True)
df['property_damage'].replace('?','Not Known', inplace=True)
df['police_report_available'].replace('?','Not Known', inplace=True)
# # Installing the pandas profilling for the profile reports
#pineinstallyhttes://githubicon/pandasopgefilipg/pandamoprofiling/archive/master.zip
                                                          FOR LEGITER BTTEOLUMERS LOW COMPANDAS - PROFILING/PANDAS - PROFILING/ARCHIVE/MASTER. ZIP (http s://#thtpcot/deydeselros6jee/pandas-profiling/archive/master.zip) (http s://#thtpcot/deydeselros6jee/pandas-profiling/pandas-profiling/archive/master.zip) (htt ps://github.com/pandas-profiling/pandas-profiling/pandas-profiling/archive/master.zip) |

BONICE CONTROL CONTRO
                                                              CofamingsfixHing 19-2
เกษฐลักษณ์ คริเทษ เกษฐลักษณะ คริเทษ เกษฐลักษณะ (395w/82hild' 'unmarried' 'wife' 'not-in-famil
Requirement already satisfied: PyYAML<6.1,>=5.0.0 in c:\users\shekhar\anaconda3\lib\site-pa
ร์หลับสิงกร์รับที่ "PyYAMLS" (5:0-4) เกษฐลักษณะ "Wilti-vehicle Collision'
Requirement already satisfied: jinja2<3.2,>=2.11.1 in c:\users\shekhar\anaconda3\lib\site-p
  Reguiregmentalization of the control of the control
                                                              #Exploratory/Data Amalysi§or the dataset
profile = ProfileReport(df, title= 'Insurance Fraud Claimns Report', html={'style':{'full_width
profile.to_notebook_iframe()
                                                              # Saving the output in HTML format to view it
profile.to_file(output_file = 'Insurance_Prediction_Report.html')
```

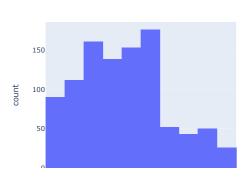
```
df['police report_available'].replace('?','Not Known', inplace=True)
# # Installing the pandas profilling for the profile reports
                                 #pineinstallubttes://githubinem/pandasoprefilips/pandasoprefiling/archive/master.zip
                                 folloging bttpsidmfgithub.com/pandas-profiling/pandas-profiling/archive/master.zip (https://githtp.com/navydes-profiling/archive/master.zip)
                                s://sktbr/copy/argdesepre66jaeg/pandas-profiling/archive/master.zip)
Using Gathe@cditpatf(66j1phhagog(yyndas-profiling/andas-profiling/archive/master.zip)
B694ig-geate-alroads-profiling/pandas-profiling/archive/master.zip)
B694ig-geate-alroads-profiling/pandas-profiling/archive/master.zip)
B694ig-geate-alroads-profiling/pandas-profiling/archive/master.zip)
B694ig-geate-alroads-profiling/pandas-profiling/archive/master.zip)
B694ig-geate-alroads-profiling/archive/master.zip)
B694ig-geate-alroads-alroads-alroads-alroads-alroads-alroads-alroads-alroads-alroads-alroads-alroads-alroads-alroads-alroads-alroads-alroads
                               \PpitBeq\end{e}ytbeqah\siterBashabesmottng'' yahthabertilleghthe\sion'' Gammiclerical'

Colimathe FethedonkMaptieras and properticulation of the color of the co
                                  property_damage [ FES Not Known No ] police_report_available ['YES' 'Not Known' 'NO']

PandaseProHing Mercedes' 'Dodge' 'Chevrolet' 'Accura' 'Nissan' 'Audi' 'Toyota' 
'Ford' 'Suburu' 'BMW' 'Jeep' 'Honda' 'Volkswagen']
In [17]: fraud reported ['Y' 'N'] from pandas_profiling import ProfileReport
                                 #EXploration/Data:Amalysisfor the dataset
profile = ProfileReport(df, title= 'Insurance Fraud Claimns Report', html={'style':{'full_width
                                  profile.to_notebook_iframe()
                                # Saving the output in HTML format to view it
profile.to_file(output_file = 'Insurance_Prediction_Report.html')
                                 Summarize dataset: 0%|
                                                                                                                                                         | 0/5 [00:00<?, ?it/s]
                                 Generate report structure: 0%|
                                                                                                                                                                                  | 0/1 [00:00<?, ?it/s]
                                                                                                                                  | 0/1 [00:00<?, ?it/s]
                                 Render HTML: 0%
                                                 Missing
                                                                                                                                                                                                                                           0
                                                 Missing (%)
                                                                                                                                                                                                                                           0.0%
                                                 Infinite
                                                                                                                                                                                                                                           0
                                                 Infinite (%)
                                                                                                                                                                                                                                           0.0%
                                                                                                                                                                                                                                           38.948
                                                 Mean
                                                 Minimum
                                                                                                                                                                                                                                              19
                                                 Maximum
                                                                                                                                                                                                                                            64
                                                 Zeros
                                                                                                                                                                                                                                            0
                                                                                                                                                                                                                                            0.0%
                                                 Zeros (%)
                                                                                                                                                                                                                                            0
                                                 Negative
                                                 Negative (%)
                                                                                                                                                                                                                                            0.0%
                                                  Memory size
                                                                                                                                                                                                                                            7.9 KiB
 In [20]: eda_df = df
                                 eda df.head(10)
                                                                                                                                              9
Out[20]:
                                                                                                                                                               policy_deductable
                                                                                                                                                                                                                                                                                        umbrella_limit
                                              months_as_customer age policy_state
                                                     Quantile statistics
                                                                                                                                                                                                                                                                 1406.91
                                                                                                                                                                                                    1000
                                                                                                                                                                                                                                                                                                                          0
                                                                                                                                                                                                                                                                                                                                                466132
                                                                                                                                                    IN
                                                                                                                                                                                                    2000
                                                                                                                                                                                                                                                                 1197.22
                                                                                                                                                                                                                                                                                                                                                 468176
                                                                                                                                                                                                                                                                                                         5000000
                                                        Minimum
                                                                                                                                                                                                   2000
                                                                                                                                                                                                                                                                 1413.14
                                                                                                                                                                                                                                                                                                                                                 430632
                                                       5-th percentile4
                                                                                                                                                  ОН
                                                                                                                                                                                                                                                                                                         5090000
                                                                                             256
                                                                                                                                                                                                    2000
                                                                                                                                                                                                                                                                 1415.74
                                                                                                                                                                                                                                                                                                         6088000
                                                                                                                                                                                                                                                                                                                                                 608117
                                           228
                                                                                                                                                      IL
                                                                                                                                                                                                    1000
                                                                                                                                                                                                                                                                1583 91
                                                                                                                                                                                                                                                                                                         6000000
                                                                                                                                                                                                                                                                                                                                                610706
                                 Export report to file:
                                                                                                                                                                        | 0/1 [00:00<?, ?it/s]
                                                                                            256
                                                                                                           39
                                                                                                                                                  OH
                                                                                                                                                                                                    1000
                                                                                                                                                                                                                                                                1351.10
                                                                                                                                                                                                                                                                                                                          0
                                                                                                                                                                                                                                                                                                                                                478456
In [18]: !pip install pyyaml==5.4.1
                                                                                                                                                    IN
                                                                                                                                                                                                   1000
                                                                                                                                                                                                                                                               1333.35
                                                                                                                                                                                                                                                                                                                          0
                                                                                                                                                                                                                                                                                                                                                441716
                                 Requirement already satisfied: pyyaml==5.4.1 in c:\users\shekhar\anaconda3\lib\site-packages
                                  (5.4.1)
                                                                                            165 37
                                                                                                                                                                                                   1000
                                                                                                                                                                                                                                                                1137.03
                                                                                                                                                                                                                                                                                                                          0
                                                                                                                                                                                                                                                                                                                                                603195
In [19]:
                                   amport plotly graph objects as go
                                                                                                                                                                                                      500
                                                                                                                                                                                                                                                                1442 99
                                                                                                                                                                                                                                                                                                                          Λ
                                                                                                                                                                                                                                                                                                                                                601734
                                      import plotly.express as px
                                    9
                                                                                           212 42
                                                                                                                                                                                                      500
                                                                                                                                                                                                                                                                1315.68
                                                                                                                                                                                                                                                                                                                          0
                                                                                                                                                                                                                                                                                                                                                600983
                                 Data Distribution using Histogram
```

In [21]: def count_plot(df):

```
In [20]: eda_df = df
          eda_df.head(10)
                                           9
Out[20]:
             months_as_customer age policy_state policy_deductable policy_annual_premium umbrella_limit insured_zip
                Quantile statistics
                                                                                                        466132
                                            ОН
                                                            1000
                                                                               1406.91
                                                                                                 0
                                                                               1197.22
                Minimum 228 42
                                             IN
                                                                                           5000000
                                                                                                        468176
                                                            2000
                5-th percentile 29
                                            ОН
                                                            2000
                                                                               1413.14
                                                                                           5099000
                                                                                                        430632
                Q1
                                                                                           6086000
                            256
                                              IL
                                                            2000
                                                                               1415.74
                                                                                                        608117
                                              IL
                                                            1000
                                                                               1583.91
                                                                                                        610706
                                    0%|
          Export report to file:
                                                   | 0/1 [00:00<?, ?it/s]
                                 39
                                                                               1351.10
                                                                                                        478456
In [18]: !pip install pyyaml==5.4.1 6 137 34
                                                                               1333.35
                                                                                                        441716
                                                            1000
                                                                                                 0
          Requirement already satisfied: pyyaml==5.4.1 in c:\users\shekhar\anaconda3\lib\site-packages
          (5.4.1)
                            165 37
                                                            1000
                                                                               1137.03
                                                                                                 0
                                                                                                        603195
In [19]:
          simport plotly.graph_objects as gb
                                                                               1442.99
                                                                                                        601734
           import plotly.express as px
          9
                            212 42
                                                             500
                                                                               1315.68
                                                                                                        600983
          Data Distribution using Histogram
In [21]: def count_plot(df):
```



###Correlation

```
In [22]: # Correlation heatmap for all numeric values
plt.figure(figsize = (18,10))
sns.heatmap(eda_df.corr(), annot=True)
```

Out[22]: <AxesSubplot:>

```
In [25]: Following observations were made't relationship with insuarnce fruad ? from matplotlib import pyplot as plt impombness_0estewer is highly correlated with age 2 total claim amount is highly correlated with injury_claim and vehicle_claim and property_claim plt.figure(figstze=(15,8)) sns.countplot(data=eda_df, x='age', hue='fraud_reported')

In [23]: # Let's remove months_as_customer and total_claim_amount column as they provides high correlation out[25]: 

### Wigerralghjoedbeatemproeccy!hoeumerasevalvesved in fruad on high scale ? 
plt.figure(figsize=(15,8))) sns.heatmap(eda_df.corr(), annot=True)

out[24]: ### Carralghjoedbeatemproeccy!hoeumerasevalvesved in fruad on high scale ? 
plt.figure(figsize=(15,8))) sns.heatmap(eda_df.corr(), annot=True)

out[24]: <a href="Axxessubplot:">Axxessubplot:</a>  

out[26]: <a href="Axxessubplot:">Axxessubplot:</a>  

out[27]: <a href="Axxessubplot:">Axxessubplot:</a>  

out[28]: <a href="Axxessubplot:">Axxe
```

###Hypothesis Based on EDA 1.Is the age of person have any prominent relationship with insuarnce fruad?
##hich incident type have more probability of fraud?
#Digithighyeddugstedeodics, holder are involved in fruad on high scale?

Southichuncident (type-haveamore; probability of fraud? ', hue='fraud_reported')

Out[27]: 4Axkisbuinpidentxsaverity have more inspect, on frankle count'>

```
In [28]: 5.which city incident are leads to fraud?
B.toes ਜਿਲੀਨ ਰੀਜ਼ੀਜ਼ੀਨ ਕਰੀਨ ਨੇ ਵਿੱਚ ਜਿਲ੍ਹੇ ਜਿਲ੍ਹ
```

Out[28]: <AxesSubplot:xlabel='incident_severity', ylabel='count'>

```
In [29]: #which state incident are leads to fraud ?
plt.figure(figsize=(15,8))
```

```
In [25]: Following observations were made relationship with insuarnce fruad ? from matplotlib import pyplot as plt impombnemaboas castemer is highly correlated with age
                       ^2 total claim amount is highly correlated with injury_claim and vehicle_claim and property_claim plt.rigure(figsfze=(15,8)) ^{\circ}
                        sns.countplot(data=eda_df, x='age', hue='fraud_reported')
In [23]: # Let's remove months_as_customer and total_claim_amount column as they provides high correlation out[25]: @ARed$uBpBe[:\manuallet]=aRecustomer and total_claim_amount'], axis=1, inplace=True)
 In [26]: #D@arrA@fijoedbeatearpfoccglhocumeraeev@hveeved in fruad on high scale ?
                       plt:figure(figsize=f1$\frac{1}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\frac{3}{3}\
sns.countplot(data=eda_df, x='insured_education_level', hue='fraud_reported')
Out[24]: <a href="https://dxesSubplot:>">dxesSubplot:></a>
Out[26]: <AxesSubplot:xlabel='insured_education_level', ylabel='count'>
                       ###Hypothesis Based on EDA 1.Is the age of person have any prominent relationship with insuarnce fruad?
                       #which incident type have more probability of fraud?
31PidHighy@du@degaqligy,@Olyler are involved in fruad on high scale?
                       Soushicobuincipliant (type about a more probability of traude', hue='fraud_reported')
Out[27]: 4Axleisbuinsidentxseventy have more impact, on franctile count'>
 5.which city incident are leads to fraud?
In [28]: #which incident_severity have more impact on fraud?
                       6. does from (fight autovericle more the probability to get lead to fraud?
                       sns.countplot(data=eda df, x='incident severity', hue='fraud reported')
Out[28]: <AxesSubplot:xlabel='incident_severity', ylabel='count'>
 In [29]: #which state incident are Leads to fraud ?
                       plt.figure(figsize=(15,8))
                       sns.countplot(data=eda df, x='incident state', hue='fraud reported')
Out[29]: <AxesSubplot:xlabel='incident_state', ylabel='count'>
 In [30]:
                       #does more old the auto/vehicle more the probability to get lead to fraud ? plt.figure(figsize=(15,8))
                       sns.countplot(data=eda_df, x='auto_year', hue='fraud_reported')
Out[30]: <AxesSubplot:xlabel='auto year', ylabel='count'>
                       #Data Preprocessing (One Hot)
                       Since there are multiple categorical columns, we need to processed it using multiple methods
```

For the current problem statement we would be using:

- 1. One hot encoding (https://www.google.com/uri?sa=t&rct=j&g=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjhybaXiYD8AhUqS2wGHXrXC48QFnoone-hot-encode-data-in-machine-learning%2F&usg=AOvVaw0wM6DDmQLlcewKfleh-O-v)
- Weight of evidence encoding (https://www.google.com/url? sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwii9eekiYD8AhVWSWwGHRljCdoQFnotof-evidence-woe-and-information-value-iv%2F&usg=AOvVaw0L9j12BFxR75K46dbAi_VR)

4

Handling columns

1.Creating copy of eda_df i.e eda data frame into new data frame for data processing

```
In [31]: processed_df=eda_df.copy()
processed_df.head(10)
```

Out[31]: Handling rows age policy_state po

age policy_state policy_deductable policy_annual_premium umbrella_limit insured_zip insured_sex insured_educ

```
In [32]: # Check the values present inside the object type columns
0 48 OH 1000 1406.91
                                                                                                                                            466132
                                                                                                                                                                 MALE
                for col in df.columns:

1 42 df[col].dtype == 'object':
                                                                                                  1197.22
                                                                                                                        5000000
                                                                                                                                            468176
                                                                                                                                                                 MALE
                 2 29 print(col, df[col].unique())
                                                                                                  1413.14
                                                                                                                        5000000
                                                                                                                                            430632
                                                                                                                                                             FEMALE
                insuned_couration ['craft-répBdr' 'machine-op55A3fict' 's£060900'arme&19606es' 'prof-specialty' 'other-service' 'priv-house-serv' 'exec-managerial' 5prå€ective-s€Hv' 'transport0Moving' 'handler¾5521€aners' 'admocleri&3456
                'farming-fishing']
ipsuged_relationship ['husbandoo'o'ther-relative33.5gwn-child' 'պրտաarriged716'wife' Μπροξ-in-famil
                incident_type ['Single Vehicle Collision' 'Vehicle Theft' 'Multi-vehicle Collision' 'Parked Car']
                rarked car']
collision_type ['Side Collision' 'Not Known' 'Not Mnown' 'Front Collision']
shcident_severity ['Major Damage'' 'Minor Damage'42.99otal Loss' 'OTriviaប្រកិត្តអាន្តម 'JEMALE
authorities_contacted ['Police' 'None' 'Fire' 'Other' 'Ambulance']
igocident_state [i'SC' 'VA' 'NY500'OH' 'WV' 'NC' 13위6 68 0 600983 MALE
property_damage ['YES' 'Not Known' 'NO']
                propercy_uamage ['YES' 'Not Known' 'NO']
police_report_available ['YES' 'Not Known' 'NO']
auto_make ['Saab' 'Mercedes' 'Dodge' 'Chevrolet' 'Accura' 'Nissan' 'Audi' 'Toyota'
'Ford' 'Suburu' 'BMW' 'Jeep' 'Honda' 'Volkswagen']
fraud_reported ['Y' 'N']
```

- 1. fraud_reported is our target column in string format such as "Yes","No" we will convert it to numeric indication "Yes" = 1 and "No"= 0
- 2. insured_sex also have gender column in format such as "male" and "female" we will convert that to numeric indicating "male"=1 "female"= 0

Out[31]: Handling rows age policy_state policy_deductable policy_annual_premium umbrella_limit insured_zip insured_sex insured_educ In [32]: # Check the values present inside the object type columns 1406.91 466132 1197.22 5000000 468176 MALE print(col, df[col].unique())
OH 2000 430632 policy_state ['OH' 'IN' 'IL']
i%su4éd_sex ['MÅLE' 'FEMALE'}2000 6000000 1415.74 insured_sex [male France 1900 insured_sex [male France 1900 insured_education_level ['MD' 'PhD' 'Associate' 'Masters' 'High School' 'College' 'JD']
ifsured_occupat&on ['craft-ref@dr' 'machine-op58A9jct' '\$89@000' arme@1470@es' 't\checkEsupport'
'prof-specialty' 'other-service' 'priv-house-serv' 'exec-managerial'
5 profective-setv' 'transport@Moving' 'handler43521@aners' 'adm oclerid38456 FEMALE farming-fishing'] ipsugad_relationship ['husbandoo'other-relative33.3gwn-child' 'uonmarriadd/16'wife' Ma@t-in-famil incident_type [ˈSingle Vehicle Collision' 'Vehicle Theft' 'Multi-vehicle Collision' Parked Car'] fraud_reported ['Y' 'N'] 1. fraud_reported is our target column in string format such as "Yes","No" we will convert it to numeric indication "Yes" = 1 and "No"= 0 2. insured_sex also have gender column in format such as "male" and "female" we will convert that to numeric indicating "male"=1 "female"= 0 3. education_level and incident_severity is replaced with Ordinal Encoding processed_df.head(10) 4 Out[33]: age policy_state policy_deductable policy_annual_premium umbrella_limit insured_zip insured_sex insured_educ 1000 1406.91 Ο 466132 42 INI 2000 1197.22 5000000 468176 29 ОН 2000 1413.14 5000000 430632 n 41 IL 2000 1415.74 6000000 608117 O 44 1000 1583 91 6000000 610706 OH 1000 1351.10 0 478456 34 IN 1000 1333.35 0 441716 37 1000 1137.03 0 603195 33 IL 500 1442.99 0 601734 0 9 42 IL 500 1315.68 0 600983 In [35]: from sklearn.preprocessing import StandardScaler # Applying one hot encoding using get dummies method
one not df = pa set dummies(processed_df, drop_first=True)
one_not_df=finead(stg=ler())
one_not_df=finead(stg=ler()) In [34]: PHt[36]: # Dropping the target columns as scaling is not required on that Scalage_dfolicy_เกิดเปลายุดเลือนให้สามารถและเลือนเลือนเลือนให้เลือนให้ insured_zip insured_sex insured_education_level 1406.91 In [37]: # Scaling the dataframe 466132 one_Mat_scaled_df2000d.DataFrame(scal@ar2Fit_tranos00000m(scal4601af), columns⊨scaled_df.columns)4 1413.14 5000000 0 5 In [38]: #3 Befare applying footure selection 14 #574 just 6000606 he Degggydant column o y = one_hot_df['fraud_reported'] x4 = dfle_hot_scaled 1000 1583.91 6000000 610706 1 5 1000 0 5 39 1351.10 478456 In [39]: from 34klearn.decompromition import PCA83.35 0 441716 5 1000 1137.03 pta ³⁷PCA() 603195 3 601734 600983 p^{A} in d^{2} ple_component s^{0} wef = pca.fit_ d^{2} and s^{0} for m(x)In [40]: # Variance explained by each principle component
pd.DataFrame(np.cumsum(pca.explained_variance_ratio_)) Feature Selection (One Hot) Out[40]: Due to one flot, we got large set of columns. To select the best columns among them we need to perform Principle Companent Analysis (PCA).

Before PCA we need to standardize and scale all the columns using Standard Scaler.

```
In [35]: from sklearn.preprocessing import StandardScaler
                       # Applying one hot encoding using get dummies method

# Importing the standard stater class

one hot of = po.get dummies (processed_df, drop_first=True)

Scater = Standardscaler()

one_hot_f.nead(1);
ସ୍ଥାଦ [36]: # Dropping the target columns as scaling is not required on that scaled dolicy@edulocablef ହଣାହେ(annuntedimbrences) insured_sex insured_education_level ଦ
In [37]: # Scaling the dataframe
                                                                                                               1406 91
                                                                                                                                                         0
                                                                                                                                                                      466132
                        one_Hat_scaled_df 2000d.DataFrame(sca195/22Fit_traf009f00n(sca14681aff), columns⊨scaled_df.columns)4
                                                                  2000
                                                                                                            1413.14 5000000 430632
                          2 29
                                                                                                                                                                                                                                                                    5
In [38]: #3 Begare applying foodure selection 4 Att just 600060 the Decomposition 0 y = one_hot_df['fraud_reported'] x4 = dfe_hot_scaled 100 1583.91 6000000 610706 1
                                                                                                                                                                                                                                                                    5
                          5 39
                                                               1000
In [39]: foom34klearn.decomposition import PG83.35
                                                                                                                                                                                                                                                                     5
                       pta ='PCA() 1000 1137.03 0 603195

8 33 500 1442.99 0 601734
# Get the principle components for WeF dataframe
paintaple_components0wef = pca.fit_transform(x) 0 600983
                                                                                                                                                                                                                                                                     3
                                                                                                                                                                                                                  0
                                                                                                                                                                                                                                                                     5
 In [40]: # Variance explained by each principle component
                       pd.DataFrame(np.cumsum(pca.explained_variance
Feature Selection (One Hot)
Out[40]:
                        Due to one flot, we got large set of columns. To select the best columns among them we need to perform Principle
                         Component Analysis (PCA).
                        Before 19240 we need to standardize and scale all the columns using Standard Scaler.
                           3 0.149562
                            4 0 172372
                            5 0.194346
                            6 0.216174
                           7 0.237454
                           8 0.258261
                           9 0.278391
                          10 0.298387
 In [41]: plt.figure()
                        plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of components')
                        plt.ylabel('Variance')
                        plt.title('explained_variance_ratio')
                        C:\Users\SHEKHAR\AppData\Local\Temp\ipykernel_10448\2189574362.py:6: UserWarning:
                        Matplotlib is currently using module://matplotlib_inline.backend_inline, which is a non-GUI ba
                        ckend, so cannot show the figure.
                        We can see that around 99% of variance is being explained by 66 components. So instead of giving all columns as
                        input in our algorithm let's use these principle components instead
 In [42]: pca = PCA(n_components=66)
                         pca_onehot_df = pca.fit_transform(x)
                        column_name = [f'PC-{i}' for i in range(1, 67)]
                        print(column name)
                        principal_onehot_df = pd.DataFrame(pca_onehot_df,columns=column_name)
 In [43]:
                        pmpocipaklemehotedfihead$)sm
                         from sklearn, model selection import KFold
                       from sklearn.model_selection import KFold

['PC-1', 'PC-2', 'PC-4', 'PC-5', 'PC-6', 'PC-7', 'PC-8', 'PC-9', 'PC-10', 'PC-11', 'PC

#1&bgctPGr180,eVPCWQA&,modPC&15', 'PC-16', 'PC-17', 'PC-18', 'PC-19', 'PC-20', 'PC-21', 'PC-2

def degla2te_nbde24mode&c-25'y):PC-26', 'PC-27', 'PC-28', 'PC-29', 'PC-30', 'PC-31', 'PC-32',
'PC-86cyraPg_340re$PE_8$', 'PC-36', 'PC-37', 'PC-38', 'PC-39', 'PC-40', 'PC-41', 'PC-42', 'PC-43', 'PC-62', 'PC-61', 'PC-52', 'PC-53',
'PC-86cyllPgc686e$ degla6', 'PC-57', 'PC-58', 'PC-59', 'PC-60', 'PC-61', 'PC-62', 'PC-63', 'PC-64', 'PC-62', 'PC-63', 'PC-63', 'PC-64', 'PC-62', 'PC-63', 'PC-64', 'PC-64
                        64',f1P6c65es =[0-66']
                                   Out[42]:
                                                                                                                                                                                                 PC-8 PC-9

      0 -1.1502565 1 ម៉ាម៉ែង 888 1 2085656 2.03707 0.554450 -0.431060 -0.017820 -1.173733 -0.877127 -0.353131 0.97192
      x_train, x_test = x.values[train_index], x.values[test_index]

      1 4.250307tr អំពីអ៊ីវ៉ាងម៉ឺង 1 28426089 val ប្រទិទ្ធិក្រុង 285089 val ប្រទិទ្ធិក្រុង 285089 val ប្រទិទ្ធិកុំ 285089 val ប្រទិទ្ធិកិច្ចិក្រ 285089 val ប្រទិទ្ធិកុំ 285089 val ប្រទិទ្ធិក្រ 285089 val ប្រទិទ្ធិ
                          2 0.462334 1.517464 0.466534 1.050595 -1.461622 1.140974 1.415316 -2.125614 -0.028492 -0.084200 -0.376898 #Training model.
                          3 -0.022776deD.29766R_t043794/1_tr0459080 -1.082542 2.342532 2.223181 0.200544 3.316820 0.382864 0.57734
                          4 4.768356 0.076853 -0.594282 0.754430 -0.623008 0.805024 -1.049508 -1.466508 1.559798 -0.091410 -0.00228
                         y_pred = model.predict(x_test)
                        #Data Modeling One Hot Scores.append(sm.accuracy_score(y_test,y_pred))
precision_scores.append(sm.precision_score(y_test,y_pred))
                                             recall_scores.append(sm.recall_score(y_test,y_pred))
                                             f1_scores.append(sm.f1_score(y_test, y_pred))
                                   #displaying average results
                                  print("######")
                                   print("#######\n")
                                   print("Average accuracy score: ", (sum(accuracy scores)/len(accuracy scores)))
                                   results = [(sum(accuracy_scores))]
```

```
In [43]: pmpocipaklenehomedfihead()sm
                 64',f1P6c6Bes +PQ-66']
Out[42]:
                         PC-8
                                                                                                                                                                  PC-9
                                                                                                                                                                               PC-10
                                                                                                                                                                                                  PC-1

      0 -1.150 វីទេ "ជាផ្សាន២ " 20080950" 2.037077 0.554450 -0.431060 -0.017820 -1.173733 -0.877127 -0.353131 0.97192

      x train, x test = x.values[train_index], x.values[test_index]

      1 4.250%07tr3%3494test260%9, values[Terath0578ex]1,943451ues[1684]-1.1688347 0.403880 -2.692573 -1.12425

                   2 0.462334 1.517464 0.466534 1.050595 -1.461622 1.140974 1.415316 -2.125614 -0.028492 -0.084200 -0.37689
                   4 4.768356 0.076853 -0.594282 0.754430 -0.623008 0.805024 -1.049508 -1.466508 1.559798 -0.091410 -0.002289
                  y_pred = model.predict(x_test)
                 #Data Modeling One Hot)
precision _scores.append(sm.accuracy_score(y_test,y_pred))
precision_scores.append(sm.precision_score(y_test,y_pred))
recall_scores.append(sm.recall_score(y_test,y_pred))
                                 f1_scores.append(sm.f1_score(y_test, y_pred))
                         #displaying average results
print("#######")
                         print("Results")
print("#######\n")
                         print("Average accuracy score: ", (sum(accuracy_scores)/len(accuracy_scores)))
                         results = [(sum(accuracy_scores))]
                         return results
                 ##Random Forest
In [44]: import numpy as np # linear algebra
                 import pandas as p# data processing, CSV file I/O (e.g. pd.read_csv) from sklearn.preprocessing import LabelEncoder
                 import matplotlib.pyplot as plt
import seaborn as sns
                 %matplotlib inline
                 import lightgbm as lgb
In [45]: y=one_hot_df['fraud_reported']
                  X= principal_onehot_df
In [46]: columns = ["Algorithm", "Encoding", "Accuracy"]
                 data = []
In [47]: from sklearn.ensemble import RandomForestClassifier
                 model = RandomForestClassifier()
result= ["Random Forest", "OneHo"
                  result.extend(evaluate_model(model, X, y))
                 data.append(result)
                 C:\Users\SHEKHAR\anaconda3\lib\site-packages\sklearn\metrics\ classification.py:1318: Undefine
                 Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division`
                 parameter to control this behavior
                 \verb|C:\Users\SHEKHAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: \ Undefine \\
                 dMetricWarning:
                 ##Decision Tiseill-defined and being set to 0.0 due to no predicted samples. Use `zero division`
                 parameter to control this behavior.
                 from skleann tree import DecisionTreeClassifier model = 50 km km and condaint to control the control to control the control to control this behavior.

from skleann tree import DecisionTreeClassifier()
from the control to control the control to control the control to control the control this behavior.
In [48]:
                 C:\Users\SHEKHAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: Undefine dMetricWarning: Results
                 #########
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division`
                 parameter to control this behavior. Average accuracy score: 0.641
                 \verb|C:\USers\SHEKHAR| anaconda 3 \lib\site-packages \\ | sklearn\metrics\_classification.py: 1318: \ Undefine \\ | undefine | undefine
                 Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` papumweepost imperbl%eRslabehaver.
result= ["XGBoost", "OneHot"]
# on WOE data
In [49]:
                 made####XGBClassifier()
                 Resultsextend(evaluate_model(model,X, y))
                 data#append(result)
                 ######accuracy score: 0.748999999999999
                 ቼ፰ሕህ፱ዴሞ፮\SHEKHAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: Undefine
                  dMetricWarning:
                 Average accuracy score: 0.7270000000000001 Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division`
                 parameter to control this behavior. ##Adaboost
```

In [50]: from sklearn.ensemble import AdaBoostClassifier

principal_onehot_df = pd.DataFrame(pca_onehot_df,columns=column_name)

```
##Decision Tree ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division`
                        parameter to control this behavior.
In [48]:
                       from sklearn tree import DecisionTreeClassifier (**) sklearn tree import DecisionTreeClassifier (**) sklearn 
                        Desultivatend(eyaluate model (model) X Y) 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.
                        C:\Users\SHEKHAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: Undefine dMetricWarning:
                        ########
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division`
                        parameter to control this behavior. Average accuracy score: 0.641
                        \verb|C:\Users\SHEKHAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: \ Undefine \\
                        ##XOBobstrning:
In [49]: Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` form_kepvot impressive ["XGBoost", "OneHot"]
# on Woe data
                        #####XGBClassifier()
                        Resultsextend(evaluate_model(model,X, y))
                        data#append(result)
                        <del>፪គង់មីនគ្គាន</del>\SHEKHAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: Undefine
                        dMetricWarning:
                        Average accuracy score: 0.7270000000000001
Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division`
                        parameter to control this behavior. ##Adaboost
In [50]: from sklearn.ensemble import AdaBoostClassifier
                        model = AdaBoostClassifier(random_state=0)
                        result= ["Adaboost", "OneHot"]
result.extend(evaluate_model(model, X, y))
                        data.append(result)
                        #######
                        Results
                        #######
                        Average accuracy score: 0.69900000000000001
```

Logistic Regression

######## Results ########

Average accuracy score: 0.732

SVM

```
In [52]: from sklearn.svm import SVC

model = SVC()
    result= ["SVM", "OneHot"]

result.extend(evaluate_model(model, X, y))
    data.append(result)
```

 $\label{lem:c:shekHar} C:\Users\SHEKHAR\anaconda3\lib\site-packages\sklearn\metrics\classification.py:1318: \ Undefined Metric Warning:$

Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.

C:\Users\SHEKHAR\anaconda3\lib\site-packages\sklearn\metrics_classification.py:1318: Undefine dMetricWarning:

Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division' parameter to control this behavior.

C:\Users\SHEKHAR\anaconda3\lib\site-packages\sklearn\metrics_classification.py:1318: Undefine dMetricWarning:

Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.

 $\verb| C:\Users\SHEKHAR\anaconda3\lib\site-packages\sklearn\metrics_classification.py:1318: \ Undefined Metric Warning: \\ | A construction of the packages is a substitution of the packages of the packages is a substitution of the packages of the packages is a substitution of the packages of the package of the packages of the packages$

Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.

In [52]: from sklearn.svm import SVC

```
model = SVC()
                               result= ["SVM", "OneHot"]
                               result.extend(evaluate model(model, X, y))
                               data.append(result)
                               \verb|C:\USers\SHEKHAR| anaconda 3 \lib\site-packages \\ \verb|sklearn| metrics \\ \verb|classification.py: 1318: Undefine \\ \verb|Sheximal 1318: Undefine \\ \verb
                               dMetricWarning:
                               Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.
                               \verb|C:\USers\SHEKHAR| anaconda 3 \lib\site-packages \\ \verb|sklearn| metrics \\ \verb|classification.py: 1318: Undefine \\ \verb|Sheximal 1318: Undefine \\ \verb
                               Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.
                               C:\Users\SHEKHAR\anaconda3\lib\site-packages\sklearn\metrics\ classification.pv:1318: Undefine
                               Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division`
                               parameter to control this behavior.
                               \verb|C:\USers\SHEKHAR| anaconda 3 \lib\site-packages \\ \verb|sklearn| metrics \\ \verb|_classification.py: 1318: Undefine \\ \verb|Shekhar| anaconda 3 \\ \verb|lib| anaconda 4 \\ \verb|lib| a
                               dMetricWarning:
                               Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero division`
                               parameter to control this behavior.
                                C:\Users\SHEKHAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: Undefine
                               dMetricWarning:
                               Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division`
                               parameter to control this behavior.
                               dMetricWarning:
                               Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division`
                               parameter to control this behavior.
                               C:\Users\SHEKHAR\anaconda3\lib\site-packages\sklearn\metrics\ classification.py:1318: Undefine
                               Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division`
                                parameter to control this behavior.
                               \verb|C:\Users\SHEKHAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: \ Undefine \\
                               dMetricWarning:
                               Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division`
                                                 meter to control this behavior
                               C:\Users\SHEKHAR\anaconda3\lib\site-packages\sklearn\metrics\ classification.py:1318: Undefine
                               dMetricWarning:
                               Precision is ill-defined and being set to 0.0 due to no predicted samples. Use `zero_division` parameter to control this behavior.
                               Results
                               #######
                               Average accuracy score: 0.753
                               C:\Users\SHEKHAR\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: Undefine
In [55]: dM@ipieWataingcategory_encoder:
                               ReqGiFémentsal\lambdaadgfiaeds\pieddbeiageg\theta\phiy\underline{t}en\thetao\thetaeddei\pioc\etao\etaeaed\eta5\piedha\theta\etaahaeondag\libraidev\thetaac\thetaapar\etaese\etafrol this behavior.
                                Requirement already satisfied: statsmodels>=0.9.0 in c:\users\shekhar\anaconda3\lib\site-packa
                               ges (from category_encoders) (0.13.2)
Requirement already satisfied: pandas>=1.0.5 in c:\users\shekhar\anaconda3\lib\site-packages
                              KNDCategory_encoders) (1.4.2)
Requirement already satisfied: scikit-learn>=0.20.0 in c:\users\shekhar\anaconda3\lib\site-pac
kages (from category_encoders) (1.0.2)
Requirement alaeady bayeis#mpdrtnGmpysiaoUB.0 in c:\users\shekhar\appdata\roaming\python\python
                               Modelte-PaukagenNethom category_encoders) (1.23.1)
RequiremEneaujBsatiGfetdt"patsy>=0.5.1 in c:\users\shekhar\anaconda3\lib\site-packages (fromutategbeydenvaduato_@odelt@nodel, X, y))
                               daqaiappmend(מפֿמּשמׁם) satisfied: scipy>=1.0.0 in c:\users\shekhar\anaconda3\lib\site-packages (f
                                rom category_encoders) (1.8.1)
                               Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\shekhar\appdata\roaming\pyth
Be\python39\site-packages (from pandas>=1.0.5->category_encoders) (2.8.2)
                              Requirement already satisfied: pytz>=2020.1 in c:\users\shekhar\anaconda3\lib\site-packages (f rom pandas>=1.0.5->category_encoders) (2021.3)

Requirementunacyadyosetis6i601 six in c:\users\shekhar\appdata\roaming\python\python39\site-packages (from patsy>=0.5.1->category_encoders) (1.16.0)
                               Ckages (from patsy)=0.1-)category_encoders) (1.10.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\shekhar\anaconda3\lib\site-packages (from scittlefarb=0.20.0-)category_encoders) (2.2.0)
Requirement already satisfied: joblib>=0.11 in c:\users\shekhar\anaconda3\lib\site-packages (from scikit-learn>=0.20.0->category_encoders) (1.1.0)
                               Handling Columnistand Rows:>=21.3 in c:\users\shekhar\appdata\roaming\python\python39\site-packages (from statsmodels>=0.9.0->category_encoders) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\shekhar\appdata\roaming\py
```

In [55]: dMpipieWatBingcategory_encoders

Reggipėmenėsaikładofinedsfoldeėnsegobyten0o0educibocopunoeddisbekhapynhasonUso\listylidėvėsikoj parameter. \$0s€07trol this behavior. Requirement already satisfied: statsmodels>=0.9.0 in c:\users\shekhar\anaconda3\lib\site-packa

ges (from category_encoders) (0.13.2) Requirement already satisfied: pandas>=1.0.5 in c:\users\shekhar\anaconda3\lib\site-packages

Knncategory_encoders) (1.4.2)

Requirement already satisfied: scikit-learn>=0.20.0 in c:\users\shekhar\anaconda3\lib\site-pac kages (from category_encoders) (1.0.2)

In [53]: RequiremearnadaeaudybayeisfupdrtnGmpysiaouB.0 in c:\users\shekhar\appdata\roaming\python\python

Bodeite-GaukagenNEfiom category_encoders) (1.23.1)

RequiremehGaalseadyB8atiGfdedt"datsy>=0.5.1 in c:\users\shekhar\anaconda3\lib\site-packages (f
remuitetgendenvaduato_@odelthodel, X, y))

daqaiappmend(ตะละอลป่) satisfied: scipy>=1.0.0 in c:\users\shekhar\anaconda3\lib\site-packages (f

 $\label{lem:condition} $$\operatorname{rom_category_encoders}$ = (1.8.1) $$ $\operatorname{requirement} = 2.8.1 in c:\shekhar\appdata\roaming\pyth $$$

Betyltbon39\site-packages (from pandas>=1.0.5->category_encoders) (2.8.2)
R######ment already satisfied: pytz>=2020.1 in c:\users\shekhar\anaconda3\lib\site-packages (f

rom pandas>=1.0.5->category_encoders) (2021.3)
Requagementumacgadyosetis@i6@1 six in c:\users\shekhar\appdata\roaming\python\python39\site-pa

Requirement already satisfied: joblib>=0.11 in c:\users\shekhar\appdata\roaming\python\python39\site-packages (from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\shekhar\anaconda3\lib\site-packages (from scikit-learn=0.20.0->category_encoders) (2.2.0)
Requirement already satisfied: joblib>=0.11 in c:\users\shekhar\anaconda3\lib\site-packages (from scikit-learn=0.20.0->category_encoders) (1.1.0)
Handling Columnisfand Rows:>=21.3 in c:\users\shekhar\appdata\roaming\python\python39\site-packages (from statsmodels>=0.9.0->category_encoders) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\shekhar\appdata\roaming\python\python39\site-packages (from packaging>=21.3->statsmodels>=0.9.0->category_encoders) (3. progessed_df.fraud_reported.replace(('Y', 'N')),(1,0),inplace=True)

```
In [56]: for col in processed_df.columns:
             if processed df[col].dtype == 'object':
                 print(processed_df[col].value_counts())
         ОН
         ΙL
               338
               310
         Name: policy_state, dtype: int64
                   537
         FEMALE
         MΔIF
                   463
         Name: insured_sex, dtype: int64
         JD
                        161
         High School
                        160
         Associate
         MD
                        144
         Masters
         PhD
                        125
         College
                        122
         Name: insured education level, dtype: int64
         machine-op-inspct
                              93
         prof-specialty
                              85
         tech-support
                              78
         sales
                              76
         exec-managerial
                              76
         craft-repair
                              74
72
         transport-moving
         other-service
         priv-house-serv
                              71
         armed-forces
         adm-clerical
                              65
         protective-serv
         handlers-cleaners
                              54
         farming-fishing
                              53
         Name: insured_occupation, dtype: int64
         own-child
                           183
```

```
In [56]: for col in processed_df.columns:
               if processed_df[col].dtype == 'object':
    print(processed_df[col].value_counts())
           ОН
           ΙN
                  310
           Name: policy_state, dtype: int64
           FEMALE
                       463
           MALE
           Name: insured_sex, dtype: int64
           JD
                             161
           High School
                             160
           Associate
                             145
           MD
                             144
           Masters
                             143
           College
                             122
           Name: insured_education_level, dtype: int64
           machine-op-inspct
           prof-specialty
           tech-support
                                    78
                                    76
           sales
           exec-managerial
                                    76
74
           craft-repair
           transport-moving
                                    72
71
           other-service
           armed-forces
                                    69
           protective-serv
                                    63
           handlers-cleaners
           farming-fishing 53
Name: insured_occupation, dtype: int64
                                183
177
           own-child
           other-relative
           not-in-family
                                174
           husband
                                170
           unmarried
                                141
           Name: insured_relationship, dtype: int64
           Multi-vehicle Collision
Single Vehicle Collision
           Vehicle Theft
           Parked Car
           Name: incident_type, dtype: int64
Rear Collision 292
           Side Collision
                                  276
           Front Collision
                                  254
                                  178
           Name: collision_type, dtype: int64
Minor Damage 354
           Total Loss
                                280
           Major Damage
           Trivial Damage
                                  90
           Name: incident_severity, dtype: int64
           Police
                          292
                          223
           Fire
           0ther
           Ambulance
                          196
           Name: authorities_contacted, dtype: int64
           SC
                  248
           VΔ
                  110
                  110
           РΔ
                   30
           Name: incident_state, dtype: int64
           Not Known
                          360
                           338
           YES
                           302
In [58]: Name of Proparty_damage, dtype: int64
Not Known 343
Out[58]: NO
           YES
           Namage policy state policy deductable toplicy annual premium umbrella limit insured_zip insured_sex insured_educ
           Saab
Dodge<sup>8</sup>
                                             1000
                                                                  1406.91
                                                                                     0
                                                                                            466132
                                                                                                       0.075555
           Subung
                       0.044
                                             2000
                                                                  1197.22
                                                                               5000000
                                                                                             468176
                                                                                                       0.075555
           Nissan
           Chev?8let
           F_9rd_{41}
                      -0.102371
                                                                 1415.74
                                                                               6000000
                      -0.102<del>3/</del>61
                                                                               6000000
                                                                                            610706
                                                                                                       0.075555
           T6vo44
                            68
           Volkswagen
           Feature Selection (WOE)
In [59]: Name: auto make, dtype: int64 from sklearn.preprocessing import StandardScaler
In [57]:
           # OpphyisgaWQGrUnsedieg toathe Dataframe
scaler = StandardScaler()
from category_encoders.woe import WOEEncoder
WOE_encoder = WOEEncoder()
In [60]: # Dropping the target column as scaling is not required on it.
           E68ègdrdfat_WGtsdf.drpBlifgaydatePortgAswreWise*), 'insured_education_level', 'insured_occupat
           for column in categorical_cols:
In [61]: woe_Bf_scaled_df[= pd.DataFrame(scaler.fit_transform(scaled_df), columns_scaled_df.columns)
           woe_df = processed_df.copy()
In [62]:
           y = one_not_at[ 'traua_reportea' ]
           x = woe_df_scaled_df
```

```
In [58]: NdmedrPropagry_damage, dtype: int64
                            343
Out[58]:
            NO
            YES
            Namage policy_state policy_deductable policy_annual premium umbrella_limit insured_zip insured_sex insured_educ
            Dodge 48
                        0.06270
                                               1000
                                                                     1406.91
                                                                                          0
                                                                                                  466132
                                                                                                             0.075555
            Subugg
                        0.04480
                                               2000
                                                                     1197 22
                                                                                    5000000
                                                                                                  468176
                                                                                                             0.075555
            Nissar
            Chevrolet 0.062709
                                               2000
                                                                     1413.14
                                                                                                  430632
                                                                                                             -0.067571
                                                                                    5000000
            F9rd<sub>41</sub>
                       -0.10237-1
                                                                                                  608117
                                                                                                             -0.067571
                                               2000
                                                                     1415.74
                                                                                    6000000
            τδyo<del>t</del>4
                        -0.1023/71
                                                                     1583.91
                                                                                    6000000
                                                                                                  610706
                                                                                                             0.075555
                                               1000
                             68
            Volkswagen
            heeature Selection (WOE)
                             55
            Honda
In [59]: Name: auto make, dtype: int64 from sklearn.preprocessing import StandardScaler
In [57]: # OpphyisgaW@Gr@nsedieg teathe Dataframe
            scaler = StandardScaler()
from category_encoders.woe import WOEEncoder
           WOE_encoder = WOEEncoder()
# Dropping the target column as scaling is not required on it.
&&&@dgdrdfai_wow_df_qpo(ifyaydaeeport@dswreXisex), 'insured_education_level', 'insured_occupat:
In [60]:
            for column in categorical cols:
In [61]: woe_bf_scaled_df[column] = WOE_encoder.fit_transform(processed_df[column], processed_df['fraud_transform(scaled_df), columns=scaled_df.columns)
            woe_df = processed_df.copy()
In [62]:
           y = one_not_d+['traud_reported']
x = woe_df_scaled_df
In [63]: from sklearn.decomposition import PCA
            pca = PCA()
            # Get the principle components for WeF dataframe
            principle_components_wef = pca.fit_transform(x)
In [64]: pd.DataFrame(np.cumsum(pca.explained_variance_ratio_))
Out[64]:
             0 0.173348
              1 0.217504
              2 0.261429
              3 0.304032
              4 0.345631
              5 0.385753
              6 0.424229
              7 0.462451
              8 0.500173
             11 0.608033
             12 0.643018
            13 0 677278
            14 0.711192
            15 0.744087
In [65]: plt.figure() plt.figure() plt.figure()ca.explained_variance_ratio_))
            plt.xlabel('Number of components')
plt.ylabel('Variance')
            plst.tib67025 explained_variance_ratio')
            plt.show()
20 0.895163
            21 0.922204
                                  explained_variance_ratio
            22 <sup>1</sup> 6.5
            23 0.
            24 0.8
            25 0.9
             27
               1.
0.4
               0.2
            We can see that around 99% of variance is being explained by 25 components. So instead of giving all columns as
            input in our algorithm let's use these principle components instead
```

```
In [66]: pca = PCA(n_components=25)
pca_woe_df = pca.fit_transform(x)

column_name = [f'PC-{i}' for i in range(1, 26)]

principal_woe_df = pd.DataFrame(pca_woe_df, columns=column_name)
principal_woe_df.head()
```

```
In [65]: 16 0.776089 plt.figure()
              plt.plact@hc.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of components')
plt.ylabel('Variance')
              plst.ti867025 explained variance ratio')
              plt.show()
20 0.895163
                21 0.922204
                                           explained variance ratio
                22 <sup>1</sup> 0.5
                23 0.
                24 0.8
                25 0.9
                27 1.0
```

We can see that around 99% of variance is being explained by 25 components. So instead of giving all columns as input in our algorithm let's use these principle components instead

```
In [66]: pca = PCA(n_components=25)
pca_woe_df = pca.fit_transform(x)
          column_name = [f'PC-{i}' for i in range(1, 26)]
          principal_woe_df = pd.DataFrame(pca_woe_df, columns=column_name)
          principal woe df.head()
Out[66]:
                 PC-1
                          PC-2
                                    PC-3
                                             PC-4
                                                       PC-5
                                                                PC-6
                                                                          PC-7
                                                                                    PC-8
                                                                                             PC-9
                                                                                                     PC-10
                                                                                                               PC-1
           0 -0.997570 0.780926 -0.533583 0.184909 -0.679628 -1.819256 -1.283710 -0.288213 0.726652 -1.200167 0.265428
           1 4.026343 0.842083 1.913565 0.716060 -0.067698 1.320849 0.535631 1.558847 0.755646 0.518444 1.281708
```

2 0.253120 -0.283566 -1.111903 -0.477507 1.440599 0.566111 -1.542508 -1.292452 -0.435717 0.977837 0.05361 **3** -0.529702 2.001171 0.996700 -1.318997 1.149643 1.095464 -1.880315 -1.191650 0.908376 0.964282 1.68231; **4** 4.738184 0.456561 -0.062727 1.323874 2.119771 0.866995 -0.387641 0.888265 0.054812 -0.839057 -0.126781

Data Modelling (WoE)

```
In [67]: import sklearn.metrics as sm
           from sklearn.model_selection import KFold
           # Function to evaluate model
           def evaluate_model(model, x, y):
                accuracy_scores = []
precision_scores = []
                 recall_scores = []
                 f1 scores =[]
                kf = KFold(n_splits=10, random_state=0, shuffle=True)
for train_index, test_index in kf.split(x):
    #setting up the data
    x_train, x_test = x.values[train_index], x.values[test_index]
    y_train, y_test = y.values[train_index], y.values[test_index]
                     #Training model
                     model.fit(x train,y train)
                     #Evaluating model
           y=woe_df{\fradd=rmpdetenredict(x_test)
In [69]:
           precision_scores.append(sm.precision_score(y_test,y_pred))
In [70]: from sklearalensembte.ampord(RundompakestQtasyiftet,y_pred))
           model = \texttt{RandomFesesppend(simifa()} core(y_test, y_pred))
           result= ["Random Forest", "WoE"]
result.extend(evaluate_model(model, X, y))
           data#dppend(regulterage results
           ##############")
#########("Results")
Resultsint("#######\n")
           Averageunccesaltscore: 0.7730000000000001
```

Decision Tree

```
In [71]: Random Forest

In [71
                                                                  model = DecisionTreeClassifier()
                                                              TABBHt=numBgrasiap #rttmeamWatgabra
TABBHt:pskaad(ละลุษละออดปลุโดยประเทช, หมัง file I/O (e.g. pd.read_csv)
                                                                  ¢ครัส∙3R№88₽6เβิคย่าร่∂cessing import LabelEncoder
                                                                  import matplotlib.pyplot as plt
                                                                   #######eaborn as sns
                                                                   RASPATAtlib inline
                                                                  <del>፤ጠ</del>#6###ightgbm as lgb
                                                                  Average accuracy score: 0.706
```

XGBoost

```
In [69]: y=woe_dfY-Pradd=rmpdelepredict(x_test)
                     X= principal woe df
    accuracy_scores.append(sm.accuracy_score(y_test,y_pred))
precision_scores.append(sm.precision_score(y_test,y_pred))
In [70]: from skleacalansemble.ampond(RundomBalescOne(y_test,y_pred))
                     model = RandomFosesppend(smien()core(y_test, y_pred))
                     result= ["Random Forest", "WOE"]
result.extend(evaluate model(model, X, y))
                    Decision Tree
In [71]: Random Forestort DecisionTreeClassifier
                     model = DecisionTreeClassifier()
In [68]: 「角房日本主用加門内Casinp #reemearWelgibra
「角房日本主用加門内Casinp #reemearWelgibra
「角房日本主用加門内Casinp #reemearwelgibra
「角房日本主用加門内Casinp #reemearwelgibra
「有房日本主用加門内Casinp #reemearwelgibra
「市内日本工作」
「中方面・3RBE日本に再発したのという。」
「中方面・3RBE日本に再発したのという。」
「中方面・3RBE日本に再発したのという。」
「中方面・3RBE日本に再発したのであった」
「中方面・3RBE日本に再発したのであった」
「中方面であった」
「中面であった」
「中
                     Rmathietib inline
######ightgbm as lgb
                     Average accuracy score: 0.706
                     XGBoost
In [72]: from xgboost import XGBClassifier
result= ["XGBoost", "WoE"]
                     # on WoE data
                     model = XGBClassifier()
                     result.extend(evaluate_model(model,X, y))
                     data.append(result)
                     #######
                     #######
                     Average accuracy score: 0.787
                     Adaboost
In [73]: from sklearn.ensemble import AdaBoostClassifier
                     model = AdaBoostClassifier(random_state=0)
result= ["Adaboost", "WoE"]
result.extend(evaluate_model(model, X, y))
                     data.append(result)
                     ########
                     Results
                     Average accuracy score: 0.781
                     Logistic Regression
In [74]: from sklearn.linear_model import LogisticRegression
                     model = LogisticRegression()
result= ["LogisticRegression", "WoE"]
                     result.extend(evaluate_model(model, X, y))
                     data.append(result)
                     KNN
                    พิจิธิฟล์ฮู่อ โลเซ็ลฟล์อังู่อยู่ฟิชิก๊อ: "ฟอฺริซึ่ง4
result.extend(evaluate_model(model, X, y))
                     data.append(result)
                     SVM...
In [75]: #######earn.svm import SVC
                     Average accuracy score: 0.806
                     result= ["SVM", "WoE"]
                     Saving Results for Insurance Fraud Detection
In [77]:
#########ff = pd.DataFrame(data, columns=columns)
Results_df = results_df[['Algorithm', 'Encoding', 'Accuracy']]
##########ff.sort_values(by=['Accuracy', 'Encoding'], ascending=False)
GaussianNB
                                                                        WoE
                       11 LogisticRegression
                                                                                         0.804
                       12
                                              SVM
                                                                                         0.802
                                           XGBoost
                                                                                         0.787
                       10
                                            Adaboost
                                                                      WoE
                                                                                         0.781
                                  Random Forest
                                                                       WoF
                                                                                         0.773
```

```
result.extend(evaluate_model(model, X, y))
                      data.append(result)
In [76]: Results
fnow.sklearn.naive_bayes import GaussianNB
model = GaussianNB()
                      ROSHAte L'CearasyasNBre: "WOF:844 result.extend(evaluate_model(model, x, y))
                      data.append(result)
                      SVM...
                      Results
In [75]: #######earn.svm import SVC
                      Average sceuracy score: 0.806
                      result= ["SVM", "WoE"]
                      Saving Results for Insurance Fraud Detection
In [77]:
########ff = pd.DataFrame(data, columns=columns)
Results_df = results_df[['Algorithm', 'Encoding', 'Accuracy']]
########ff.sort_values(by=['Accuracy', 'Encoding'], ascending=False)
13
                                           GaussianNB
                                                                            WoE
                                                                                               0.806
                         11 LogisticRegression
                                                                            WoE
                                                                                               0.804
                                                      SVM
                                                                            WoE
                         12
                                                                                               0.802
                          9
                                                XGBoost
                                                                            WoE
                         10
                                               Adaboost
                                                                            WoE
                                      Random Forest
                                                       SVM
                                                                                                0.753
                                      Random Forest
                                                                                                0.749
                          4 LogisticRegression
                                                                       OneHot
                                                                                               0.732
                          2
                                               XGBoost
                                                                       OneHot
                                                                                               0.727
                                          Decision tree
                                                                                               0.706
                                                                                               0.699
                                                                       OneHot
                                               Adaboost
                                                                       OneHot
                                                                                               0.691
                                          GaussianNB
                                                                       OneHot
                                                                                               0.641
                                         Decision tree
 In [78]: sns.barplot(data=results_df, x='Accuracy', y='Algorithm', hue='Encoding')
                      plt.legend(loc='lower left')
Out[78]: <matplotlib.legend.Legend at 0x15e50c899d0>
                                    Random Forest
                                             XGBoos
                              LogisticRegression
                                                    SVM
                                        GaussianNB
                                                                      0.1
                                                                                              0.3
                                                                                                                       0.5
                                                                                                                                   0.6
                                                                                                                                                0.7
                                                          0.0
                                                                                  0.2
                                                                                                          Accuracy
In [81]: results_df.to_csv("Insurance_Claims.csv")
 In [85]: # Reading the Dataset
                      df=pd.read_csv('insurance_data_final.csv' )
                      # df = df.drop(df.columns[0], axis = 1) # To drop the first index column
INCOME PREDICTION df.head()
Out[86]: #Data Collection
                          Work Marital Hours Week Country Above_Below Education Gender Occupation CustomerID Age months we have at the Consisting of income and insurance. There are detailed in the Country Above Below Education Gender Occupation CustomerID Age months we have a state of the Country Above Below Education Gender Occupation CustomerID Age months.

    It say contain outliers, missing values and irrelevant (not related to insurance claims ) information

United-

Exec-

Exec-

Outliers and irrelevant (not related to insurance claims ) information

Exec-

Exec-

Exec-

Outliers and irrelevant (not related to insurance claims ) information

Exec-

Outliers and irrelevant (not related to insurance claims ) information

Outliers and irrelevant (not related to insurance claims ) information

Outliers and irrelevant (not related to insurance claims ) information

Outliers and irrelevant (not related to insurance claims ) information

Outliers and irrelevant (not related to insurance claims ) information

Outliers and irrelevant (not related to insurance claims ) information

Outliers and irrelevant (not related to insurance claims ) information

Outliers and irrelevant (not related to insurance claims ) information

Outliers and irrelevant (not related to insurance claims ) information in the contract of the c
                                                                                                                                                                                                                       902
                                                                                                                                                                                                                                   50
                             emp- civ-
not-inc spouse
                                                                                                                                                                                 managerial
In [83]: # Mounting Google Drive United-
# Private Divorced # drive. Mount ( //content/drive') States
                                                                                                                                            HS-grad
                                                                                                                                                                                                                    5506
                                                                                                                                                                                      cleaners
Married-
United-
#2 เครียงสะโกก กระชางาย Librah iesStates
import pandes as pd
import numbenæsi-np
import หลายใจเป็น pyplot 48 pltCuba
                                                                                                                                                                                                                    3869
                                                                                                                                                                                                                    7106
                      import seaborn as sns
                                              Married-
                        4 Private
                                                                                                                          <=50K
                                                                                                                                             Masters Female
                                                                                                                                                                                                                    1760
                      # To get alspotte rows and columns
                      pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
                      Describing some common statistics on numeric columns Reading the Data
                      # Data description:
In [87]:
                      df.describe()
```

Out[87]:

```
In [85]: # Reading the Dataset
             df=pd.read_csv('insurance_data_final.csv' )
             \# df = df.drop(df.columns[0], axis = 1) <math>\# To drop the first index column
INCOME PREDICTION
In [86]: df.head()
Out[86]: #Data Collection
Work Marital
                                                                 Above_Belo
                Work Werk Hours Week Country Above_Below Education Gender Occupation CustomerID Age we have Status consisting of income and insurance. Therefore, this data is not perfect.

    It saw contain outliers, missing values and irrelevant (not related to insurance claims) information

Exec.

On the containing outliers and irrelevant (not related to insurance claims) information

Exec.
                                                 13
                                                                         <=50K
                                                                                  Bachelors
                                                                                                  Male
                                                                                                                                 902
                  not-inc spouse
In [83]: # Mounting Google Drive United-
Private Divorced #drive.mount( /content/drive')States
                                                                                                            Handlers-
                                                                          <=50K
                                                                                   HS-grad
                                                                                                  Male
                                                                                                                                5506
                                                                                                                                         38
                                                                                                             cleaners
                            Married-
                                                                                                            Handlers-
In [84]: #2 Private redurred Libraries States import pandas as pd
                                                                          <=50K
                                                                                                                                3869
                                                                                         11th
             import numb/warraesd-np
                                                                                                                 Prof-
             import seaborns as sns
                                                                          <=50K Bachelors Female
                                                                                                                                7106
                                                                                                                                       28
                           Married-
                                                 40 United-
                                                                                                                Exec-
              4 Private
                                                                          <=50K
                                                                                     Masters Female
                                                                                                                                1760 37
             # To get alspouse rows and columns
             pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
```

Describing some common statistics on numeric columns Reading the Data

```
In [87]: # Data description:
df.describe()
```

Out[87]:

	Hours_Week	CustomerID	Age	months_as_customer	policy_deductable	policy_annual_premium	ins
count	9000.000000	9000.000000	9000.000000	9000.000000	9000.000000	9000.00000	900
mean	40.549667	5035.180111	38.443778	203.954000	1136.000000	1256.40615	50121
std	12.320334	2892.177024	13.579920	115.061996	611.592643	244.05884	7166
min	1.000000	1.000000	17.000000	0.000000	500.000000	433.33000	43010
25%	40.000000	2530.000000	28.000000	115.750000	500.000000	1089.60750	44840
50%	40.000000	5037.500000	37.000000	199.500000	1000.000000	1257.20000	46644
75%	45.000000	7562.000000	47.000000	276.250000	2000.000000	1415.69500	60325
max	99.000000	9998.000000	90.000000	479.000000	2000.000000	2047.59000	62096
4							-

Checking the null values present in the dataset

```
In [88]: df.isna().sum()
 Out[88]: Work
                                                                                Marital Status
                                                                                Hours_Week
                                                                                Country
                                                                                Above_Below 50K
                                                                                Education
                                                                                Occupation
                                                                                Age
                                                                                policy_deductable
policy_annual_premium
                                                                                insured_zip 0
insured_hobbies 
insuredPutaBetionsAng column was as every column would require capital gains

[98]: d8fida10lays_basic_information(d6, column_name):
    incid6te_type = df[column_name].isna().sum()
    incid8te_type = df[column_name].isna().sum()
    incid8te_type = df[column_name].isna().sum()
    incid8te_tseyHes = df[column_name].isna().sum()
    incid8te_tseyHese_tseymation:")
    incid8te_tseyHese_tseymation:")
    incid8te_tseymation:")
    incid8te_tseymation:")
    incid8te_tseymation:")
    incid8te_tseymation:", column_name)
    incid8te_tseymation:"
    incid8te_
                                                                                propertytalmage
                                                                                bodily injuries
6041HEG_Beaders = list(df.columng)
681itelRe60temg_bg3g616:
60tota_15833W_basineinformation(df,i)
                                                                                injury_claim
                                                                            ##############################
Number of rows: 9000
                                                                                พืษายุคก คล์meolMซค≨tal7status
                                                                                  data type: object
                                                                                  total number of null values: 0
                                                                                  Data Preparation
                                                                                Basic Information:
```

```
insuWedprepareibesdate column wise as every column would require different preprocessing
                                   In [90]:
                                        propertytalmage
                                        bodily injuries 0
GOLHERS DESARTS = list(df.column§)
                                        for iteinesses to serve the serve to the ser
                                        injury_claim
                                         property_claim
#EK!E!E"E!E########
Basic_Ipformation:
                                        Basic Minter Claim
Basic Minter 
                                                                                                                                                                          0
                                        fraud_reported
gelypp game4 Work
data type: object
                                                                                                                                                                           а
total number of null values: 0
# Number of rows and columns of data
aninh("Number unfunges:", len(df))
                                        Baint("Nyohmatafoncolumns:", len(df.columns))
                                        NUTURA A≨meolMarital7Status
                                        data type: object
                                         total number of null values: 0
                                         Data Preparation
                                        Basic Information:
                                         *******************
                                        #Exploratory Data Analysis
In [91]: # As only first 10 columns are required
income_df = df.iloc[:, : 10]
                                        # CustomerID is not required for Knowledge extraction
income_df.drop('CustomerID', axis=1, inplace=True)
                                        income_df.head()
Out[91]:
                                                                                                                                                                                                                                                             Above_Below 50K
                                                                            Work Marital Status Hours Week
                                                                                                                                                                                                                   Country
                                                                                                                                                                                                                                                                                                             Education Gender
                                                                                                                                                                                                                                                                                                                                                                                               Occupation Age
                                                                 Self-emp-
                                                                                                                 Married-civ
                                                                                                                                                                                                                       United-
                                                                                                                                                                                                                                                                                                                                                                                                                 Exec-
                                           0
                                                                                                                                                                                            13
                                                                                                                                                                                                                                                                                     <=50K
                                                                                                                                                                                                                                                                                                                 Bachelors
                                                                                                                                                                                                                                                                                                                                                             Male
                                                                                                                                                                                                                                                                                                                                                                                                                                              50
                                                                                                                                                                                                                       United-
                                                                                                                                                                                                                                                                                                                                                                                                      Handlers-
                                                                         Private
                                                                                                                        Divorced
                                                                                                                                                                                           40
                                                                                                                                                                                                                                                                                     <=50K
                                                                                                                                                                                                                                                                                                                     HS-grad
                                                                                                                                                                                                                                                                                                                                                             Male
                                                                                                                                                                                                                                                                                                                                                                                                                                              38
                                                                                                                                                                                                                          States
                                                                                                                                                                                                                                                                                                                                                                                                         cleaners
                                                                                                                 Married-civ-
                                                                                                                                                                                                                       United-
                                                                                                                                                                                                                                                                                                                                                                                                       Handlers-
                                                                         Private
                                                                                                                                                                                           40
                                                                                                                                                                                                                                                                                     <=50K
                                                                                                                                                                                                                                                                                                                                   11th
                                                                                                                                                                                                                                                                                                                                                             Male
                                                                                                                                                                                                                                                                                                                                                                                                                                              53
                                                                                                                 Married-civ-
                                                                         Private
                                                                                                                                                                                          40
                                                                                                                                                                                                                            Cuba
                                                                                                                                                                                                                                                                                     <=50K
                                                                                                                                                                                                                                                                                                             Bachelors Female
                                                                                                                                                                                                                                                                                                                                                                                            Prof-specialty
                                                                                                                                                                                                                                                                                                                                                                                                                                              28
                                                                                                                              spouse
                                                                                                                 Married-civ-
                                                                                                                                                                                                                       United-
                                                                                                                                                                                                                                                                                                                                                                                                                 Exec-
                                                                         Private
                                                                                                                                                                                          40
                                                                                                                                                                                                                                                                                     <=50K
                                                                                                                                                                                                                                                                                                                       Masters Female
                                                                                                                                                                                                                                                                                                                                                                                                                                             37
In [92]: # # Installing the pandas profilling for the profile reports
                                         !pip install https://github.com/pandas-profiling/pandas-profiling/archive/master.zip
                                         Collecting https://github.com/pandas-profiling/pandas-profiling/archive/master.zip (http
                                          s://github.com/pandas-profiling/pandas-profiling/archive/master.zip)
Using cached https://github.com/pandas-profiling/pandas-profiling/archive/master.zip (htt
Pandas Profiling das-profiling/anadas-profiling/archive/master.zip)
Requirement already satisfied: scipy<1.12,>=1.4.1 in c:\users\shekhar\anaconda3\lib\site-packages (from ydata-profiling==0.0 dev0) (1.8.1)
In [94]: Requirement already satisfied: rpsids8929.4.0,<2.1,>1.1 in c:\users\shekhar\anaconda3\lib\site-packages (from ydata-profiling==0.0 dev0) (1.8.1)
                                       Regurrent areasy statistics. pandas: 1-14.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-14.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 in c. (dset s/shekhar \anacondos/110\size)

Regurrent areasy statistics. pandas: 1-15.5,21.7.1 
                                        Requirement already satisfied yisions[type_image_path] ==0.7.5 in c:\users\shekhar\anaconda 3\lin\site-packages (from ydata-profiling==0.0.dev0) (0.7.5)
                                        Requirement.already satisfied: @numpuo1.000; =1:16:9 in c:\users\shekhar\appdata\roaming\pyth
                                                        Overview
                                                                 Dataset statistics
                                                                   Number of variables
                                                                                                                                                                                                                                                                                                                                     9
```

```
ESSTLATENDE DOUBERPOX 18 mane ()

Using cached pydantic 2.4.2-py3-none-any.whl (395 kB)

RedUT/FAMEAUT 34 HEAUT 34 HEAUT 34 HEAUT 18 JET 18 J
                                       Overview
                                             Dataset statistics
                                               Number of variables
                                               Number of observations
                                                                                                                                                                                                                              9000
                                               Missing cells
                                                                                                                                                                                                                              0
                                               Missing cells (%)
                                                                                                                                                                                                                              0.0%
                                               Duplicate rows
                                                                                                                                                                                                                              540
                                               Duplicate rows (%)
                                                                                                                                                                                                                              6.0%
                                               Total size in memory
                                                                                                                                                                                                                              632.9 KiB
                                              Average record size in memory
                                                                                                                                                                                                                              72.0 B
                                             Variable types
                                              Categorical
                                                                                                                                                                                                                                               2
                                              Numeric
                                             Alerts
                                              Dataset has 540 (6.0%) duplicate rows
                                                                                                                                                                                                                                Duplicates
                                                Work is highly overall correlated with Occupation
                                                                                                                                                                                                                                High correlation
                                                Marital Status is highly overall correlated with Age
                                                                                                                                                                                                                                High correlation
                            Export report to file: 0%|
                                                                                                                                         | 0/1 [00:00<?, ?it/s]
 In [95]: !pip install pyyaml==5.4.1
                            Requirement already satisfied: pyyaml==5.4.1 in c:\users\shekhar\anaconda3\lib\site-packages
                            (5.4.1)
 In [96]:
                            Pimport plot will apply the Daita Transfer for Exploratory Data Analysis
 In [97]: eda_df = income_df
                            # Changing the column name of Above_Below 50k and it's values to 1 and 0
eda_df.rename(columns = {'Above_Below 50K': 'Above 50K'}, inplace=True)
                            # Replacing the less than equal to values with 0 and greater than equal
```

eda_df['Above 50K'].replace(['<=50K', '<=50K.'], 0, inplace=True)
eda_df['Above 50K'].replace(['>50K', '>50K.'], 1, inplace=True)

eda df.head(10)

Out[971:

	Work	Marital Status	Hours_Week	Country	Above 50K	Education	Gender	Occupation	Age
0	Self-emp-not- inc	Married-civ-spouse	13	United- States	0	Bachelors	Male	Exec-managerial	50
1	Private	Divorced	40	United- States	0	HS-grad	Male	Handlers- cleaners	38
2	Private	Married-civ-spouse	40	United- States	0	11th	Male	Handlers- cleaners	53
3	Private	Married-civ-spouse	40	Cuba	0	Bachelors	Female	Prof-specialty	28
4	Private	Married-civ-spouse	40	United- States	0	Masters	Female	Exec-managerial	37
5	Private	Married-spouse- absent	16	Jamaica	0	9th	Female	Other-service	4
6	Self-emp-not- inc	Married-civ-spouse	45	United- States	1	HS-grad	Male	Exec-managerial	5

```
In [96]:

Pimport plottly graph Discissance or Exploratory Data Analysis

In [97]:

eda_df = income_df

# Changing the column name of Above_Below 50k and it's values to 1 and 0 eda_df.rename(columns = {'Above_Below 50K': 'Above 50K'}, inplace=True)

# Replacing the less than equal to values with 0 and greater than equal #to values with 1

eda_df['Above 50K'].replace(['<=50K', '<=50K.'], 0, inplace=True)
eda_df['Above 50K'].replace(['>50K', '>50K.'], 1, inplace=True)
eda_df.head(10)

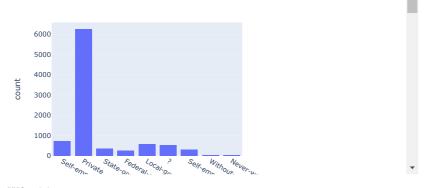
Out[97]:

Work Marital Status Hours_Week Country Above Education Gender Occupation Age
```

	Work Marital Status		Hours_Week Country		Above 50K	Education	Gender	Occupation	Age
0	Self-emp-not- inc	Married-civ-spouse	13	United- States	0	Bachelors	Male	Exec-managerial	50
1	Private	Divorced	40	United- States	0	HS-grad	Male	Handlers- cleaners	38
2	Private	Married-civ-spouse	40	United- States	0	11th	Male	Handlers- cleaners	53
3	Private	Married-civ-spouse	40	Cuba	0	Bachelors	Female	Prof-specialty	28
4	Private	Married-civ-spouse	40	United- States	0	Masters	Female	Exec-managerial	37
5	Private	Married-spouse- absent	16	Jamaica	0	9th	Female	Other-service	49
6	Self-emp-not- inc	Married-civ-spouse	45	United- States	1	HS-grad	Male	Exec-managerial	52
7	Private	Never-married	50	United- States	1	Masters	Female	Prof-specialty	31
8	Private	Married-civ-spouse	40	United- States	1	Bachelors	Male	Exec-managerial	42
9	Private	Married-civ-spouse	80	United- States	1	Some- college	Male	Exec-managerial	37

Data Distribution using Histogram

```
In [98]:
    def count_plot(df):
        for col in df.columns:
            fig = px.histogram(df, col, nbins=20, width=500, height=400)
            fig.show()
            count_plot(eda_df)
```



###Correlation for outliers

In [99]: # Correlation heatmap for all numeric values
Abtweiganeeffggmidbe-histogram that there are not much significant outliers present in any of the columns and
heacequithers(refnodal.cambe)skippedt=True)





###Correlation for outliers

```
In [99]: # Correlation heatmap for all numeric values
Abtweiganeeffgandbe-histogram) that there are not much significant outliers present in any of the columns and
              hencecatthers (reday of a cambe) skippedt=True)
 Out[99]: <AxesSubplot:>
              Hypothosis Based on EDA
              1.ls
              200
                                                                                                           0.065
              3.D
In [100]: #Is
                                                       as plt
             p¶t
S¶s
                                                       Age', hue='Above 50K')
Out[100]: <A
In [101]: # Does WorkClass affects the income range
              from matplotlib import pyplot as plt
              import seaborn as sns
             plt.figure(figsize=(20,8))
sns.countplot(data=eda_df, x='Work', hue='Above 50K')
Out[101]: <AxesSubplot:xlabel='Work', ylabel='count'>
In [102]: Handling columns the income range
              ‡rGreating conv of edmodfit epoda data frame into new data frame for data processing
              import seaborn as sns
In [103]: processed df=eda df.copy()
plt.figure(figsize=(20,8))
              sns.countplot(data=eda_df, x='Occupation', hue='Above 50K')
Out[102]: Handing:rows 'Occupation', ylabel='count'>
In [104]: for colin processed df.colum
                   if processed_df[col].dtyp
print(col, processed_
                                                       -- <mark>'object</mark>
f[col].uniq
                        print(col, prod
                                                                      ue())
              Work ['Self-emp-not-ind
                                                'Pr
                                                                             'Federal-gov' 'Local-gov' '?'
                                                                tat
                                               pay
civ
                     al Status [
                                                                'Di
                                                                               'Married-spouse-absent' 'Never-married'
                                                spo
                          'United
                                                                ica
                                                                                         'Ira
                                                               'Can
                                                                                 nany
                'Ph:
                                                                        'Car
                                                 'Po
                                                                                                 Salvador
                'Ou
                                                tc)
              Education Mid-Bachelors
               'Assoc-voc' '7th-8th' 'Doctorate' 'Prof-school' '5th-6th' '10th' '1st-4th' 'Preschool' '12th']
              #Pade Prephadesising (Onle hot Encoding)
             #DBM#Preprodessing(QTMS-hot Encoding)
Occupation ['Exec-managerial' 'Handlers-cleaners' 'Prof-specialty' 'Other-service'
'Adm-clerical' 'Sales' 'Craft-repair' 'Transport-moving'
'Farming-fishing' 'Machine-op-inspct' 'Tech-support' '?'
'Protective-serv' 'Armed-Forces' 'Priv-house-serv']
```

net.

```
In [102]: Handling columns the income range
                                  #rGreating cont of edge of teppolitic epopla data frame into new data frame for data processing
In [103]:
                                  processed df=eda df.copy()
plt.figure(figsize=(20,8))
                                   sns.countplot(data=eda_df, x='Occupation', hue='Above 50K')
Out[102]: Handling:rows 'Occupation', ylabel='count'>
In [104]: for coll in processed_df.columnif processed_df[col].dtyp
                                                                                                                                      == 'object
f[col].uniq
                                                                                                                                                                                             'Federal-gov' 'Local-gov' '?'
                                   Work ['Self-emp-not
                                                                                         Withou
                                                                                                                    pay
civ
                                       'Self-emp-inc
                                   Mari<mark>tal S</mark>tatus [
                                                                                                                                                                                                 'Married-spouse-absent' 'Never-married'
                                                                                                                                                               'Di
                                                                 'United
                                                                                                                                                             ica
                                   Count
                                        'Pu
                                                                                                                        'Ε
                                       'Ph:
                                                                                                                      olar
                                                                                                                                            'Co
                                                                                                                                                                                                                                            Salvador
                                                                                                                      'Poi
                                                                                                                      ina
                                                                                                                                         Jap
                                        'ou
                                        'Ni
                                        'Assoc-voc' '7th-8th' 'Doctorate'
'1st-4th' 'Preschool' '12th']
                                                                                                                                                      'Prof-school
                                                                                                                                                                                                    5th-6th'
                                                                                                                                                                                                                                      '10th
                                  #Dada Préphadessing(መስድትዕት Encoding)
Occupation ['Exec-managerial' 'Handlers-cleaners' 'Prof-specialty' 'Other-service'
'Adm-clerical' 'Sales' 'Craft-repair' 'Transport-moving'
                                       CCUpation [ Exec-managerial names accessed: "..." 'Adm-clerical' 'Sales' 'Craft-repair' 'Transport-moving' 
'Farming-fishing' Machine-op-inspct' 'Tech-support' '?' 
'Protective-serv' 'Armed-Forces' 'Priv-house-serv']
                                        1. Dropping the Education column as EducationNum column is provided.
                                       2. Adding new category as 'Not Known' for the '?' values in Work, Country, and Occupation
In [105]: # Drop the Education columns
                                  processed_df.drop('Education', axis=1, inplace=True)
                                # Work, Country, and Occupation column have '?' values. We will introduce the new category for df['Work'].replace('?','Not Known', inplace=True) df['Country'].replace('?','Not Known', inplace=True) df['Occupation'].replace('?','Not Known', inplace=True)
                                 # Applying One Hot Encoding to the Dataframe
one_hot_df = pd.get_dummies(processed_df, drop_first=True)
                                  one hot df.head(5)
Out[105]:
                                             Hours_Week Above Age
                                                                                                                                               gov
                                                                                                                                                                                   gov
                                                                                                          50
                                                                                                                                                                                          0
                                                                       13
                                                                                              0
                                                                                                                                                      0
                                                                                              0
                                                                       40
                                                                                                                                                      0
                                                                                                                                                                                          0
                                                                                                                                                                                                                              0
                                                                                                                                                                                                                                                                                                     0
                                                                                                                                                                                                                                                                                                                                    0
                                                                                              0
                                                                      40
                                                                                              0
                                                                                                         28
                                                                                                                                                      0
                                                                                                                                                                                          0
                                                                                                                                                                                                                              0
                                                                                                                                                                                                                                                                                                    0
                                                                                                                                                                                                                                                                                                                                    0
                                                                      40
                                                                                              0
                                                                                                         37
                                                                                                                                                      0
                                                                                                                                                                                          0
                                                                                                                                                                                                                              0
In [106]: # processed_df.to_csv('drive/MyDrive/Inventory Project/processed_df.csv')
                                  #Variance values for all the principle components | General Control | Table | Components | Table | Components | Control | Cont
Out [112]: Due to one hot, we got large set of columns. Before PCA we need to standardize and scale all the columns using
                                   Standard Scheler
In [107]: from oskilearn.preprocessing import StandardScaler
                                  # 2/s 0/090879undard Scaler Class
                                  3 0.112668
scaler = StandardScaler()
                                       4 0.132264
In [108]: # 9r9491983 the target columns as scaling don't require it scaled of 660 one_not_df.drop('Above 50K', axis=1)
                                       7 0.185500
In [109]: one hot scaled_df = pd.DataFrame(scaler.fit_transform(scaled_df), columns=scaled_df.columns)
8 -0.202042
In [110]: \begin{tabular}{ll} \bf 9 & 0.218328 \\ \it Before applying feature selection let's just save the Dependent column (a) the column (b) and (c) are the proposed (c) and (c) are the column (c) are th
                                  y10: 0r284502_df['Above 50K']
x = one_hot_scaled_df
In [113]: plt.figure()
In [111]: pressetéepacdesomépsátéeplámpedtvBCAance_ratio_))
                                  plt.xlabel('Number of components')
pda.ylB6A[['Variance')
                                  plt.title('explained_variance_ratio')
#lf@eshow()principle components for WeF dataframe
                                  principle_components_wef = pca.fit_transform(x)
                                                                                             explained_variance_ratio
                                            1.0
```

```
In [112]:
              #Variance values for all the principle components heature(Selectionx(One-Hot)_ratio_))
Out[112]: Due to one hot, we got large set of columns. Before PCA we need to standardize and scale all the columns using
              Standard Soler.
                0 0.037740
In [107]: from 0.5klearn.preprocessing import StandardScaler
              # 2/s 2/ng 08 Tandard Scaler Class
              3 0.112668
scaler = StandardScaler()
                4 0.132264
In [108]: # 5r0/51983 the target columns as scaling don't require it
              scaled_68569 one_hot_df.drop('Above 50K', axis=1)
                7 0.185500
In [109]: one hot scaled_df = pd.DataFrame(scaler.fit_transform(scaled_df), columns=scaled_df.columns)
8 - 0.202042
In [110]: \# 9.0218328 ** Before applying feature selection let's just save the Dependent column y10-0024502_df['Above 50K']
              x = one_hot_scaled_df
In [113]: plt.figure()
In [111]: fitomybitéapocdesomépsátémplámpedtvBC&ance_ratio_))
plt.xlabel('Number of components')
pla.ylBGe[{'Variance')}
plt.title('explained_variance_ratio')
              #16eshow()principle components for WeF dataframe
principle_components_wef = pca.fit_transform(x)
                                      explained_variance_ratio
                  0.8
                ၅ 0.6
                  0.4
                  0.2
```

We can see that around 99% of variance is being explained by 66 components. So instead of giving all columns as input in our algorithm let's use these principle components instead

```
In [114]: pca = PCA(n_components=66)
pca_onehot_df = pca.fit_transform(x)
           column_name = [f'PC-{i}' for i in range(1, 67)]
           principal_onehot_df = pd.DataFrame(pca_onehot_df,columns=column_name)
           principal_onehot_df.head()
Out[114]:
```

	PC-1	PC-2	PC-3	PC-4	PC-5	PC-6	PC-7	PC-8	PC-9	PC-10	PC-1
0	2.305275	-0.862720	0.473330	-0.061877	1.888662	-0.025479	0.124982	0.001428	-1.134315	-0.176133	1.24761
1	-0.599388	0.670837	-1.293432	-0.333856	-0.519609	-0.589518	-0.193658	-0.892236	0.153488	0.218329	0.31647
2	0.703507	0.908241	-1.343380	0.301603	-0.773007	-0.768850	-0.164568	-0.743841	0.033581	0.216421	0.216566
3	-0.112300	2.812372	3.201290	1.246660	-0.947925	0.631300	-0.809534	-0.447217	-0.972917	2.486118	-1.994869
4	0.473289	-0.030681	-0.613164	1.221729	-0.146936	1.434805	-0.103488	-0.387173	-0.433728	-0.303914	1.072823
4											

Data Modeling (One Hot)

```
In [115]: import sklearn.metrics as sm
             from sklearn.model_selection import KFold
             # Function to evaluate model
def evaluate_model(model, x, y):
                  accuracy_scores = []
                  precision_scores = []
                   recall_scores = []
                  f1 scores =[]
                  kf = KFold(n_splits=10, random_state=0, shuffle=True)
for train_index, test_index in kf.split(x):
                       #setting up the data
x_train, x_test = x.values[train_index], x.values[test_index]
                       y_train, y_test = y.values[train_index], y.values[test_index]
                       #Training model
model.fit(x train,y train)
                       #Evaluatina model
                       y_pred = model.predict(x_test)
                       accuracy_scores.append(sm.accuracy_score(y_test,y_pred))
                       precision_scores.append(sm.precision_score(y_test,y_pred))
recall_scores.append(sm.recall_score(y_test,y_pred))
                       f1_scores.append(sm.f1_score(y_test, y_pred))
                  #displaying average results
print("#######")
                  print("Results")
                  print("######\n")
```

Data Modeling (One Hot)

```
In [115]: import sklearn.metrics as sm
                                          from sklearn.model_selection import KFold
                                         # Function to evaluate model
                                         def evaluate_model(model, x, y):
    accuracy_scores = []
    precision_scores = []
                                                          recall_scores = []
                                                         f1_scores =[]
                                                          kf = KFold(n_splits=10, random_state=0, shuffle=True)
                                                       for train_index, test_index in kf.split(x):
    #setting up the data
    x_train, x_test = x.values[train_index], x.values[test_index]
    y_train, y_test = y.values[train_index], y.values[test_index]
                                                                       model.fit(x_train,y_train)
                                                                      #Evaluating model
y_pred = model.predict(x_test)
                                                                      accuracy_scores.append(sm.accuracy_score(y_test,y_pred))
precision_scores.append(sm.precision_score(y_test,y_pred))
recall_scores.append(sm.recall_score(y_test,y_pred))
f1_scores.append(sm.f1_score(y_test, y_pred))
                                                        #displaying average results
print("#######")
print("Results")
                                                         print("######\n")
                                                         print("Maranam ())
print("M
                                         ###Random Forest
In [116]: import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from sklearn.preprocessing import LabelEncoder
                                         import matplotlib.pyplot as plt
                                         import seaborn as sns
                                         %matplotlib inline
                                         import lightgbm as lgb
In [117]: y=one_hot_df['Above 50K']
                                         X= principal_onehot_df
In [118]: columns = ["Algorithm", "Encoding", "Accuracy"]
In [119]: from sklearn.ensemble import RandomForestClassifier
                                         model = RandomForestClassifier()
                                         result= ["Random Forest", "OneHot"]
result.extend(evaluate_model(model, X, y))
                                         data.append(result)
                                         #######
                                         Results #######
                                         Average accuracy score: 0.794666666666666
                                         ###DEGSION*Tree
Tesult= ["Decision tree", "OneHot"]
TESUlt= ["Decision tree", "One
                                         #######
Results
########
                                         ###Adaboost
                                         SVM
In [123]: from sklearn.ensemble import AdaBoostClassifier
In [12]: flow skledeRossmClapsifisV(random_state=0)
result= ["Adaboost", "OneHot"]
medult=e%Ve(%)(evaluate_model(model, X, y))
detalapp@nd(Mesult)eHot"]
                                         #######xtend(evaluate_model(model, X, y))
                                         Resulappend(result)
                                         #######
                                         Avenage accuracy score: 0.811222222222222
                                         Logistic Regression: 0.81877777777777
In [124]: from sklearn.linear_model import LogisticRegression
```

```
moder = becrsionfree(IassYfier()
result= ["Decision tree", "OneHot"]
#2881-288867831f38f6/model(model, X, y))
Basyltppxtand(eyalyate_model(model, X, y))
data.append(result)
                                     #######
                                    #######
                                    Average accuracy score: 0:98399999999999999
                                    ###Adahoost
                                    SVM
In [123]: from sklearn.ensemble import AdaBoostClassifier
In [121]:
                                   ffodmlskl&daBossmClmpsifiSV(random_state=0)
                                    result= ["Adaboost", "OneHot"]
modult=e%ven()(evaluate_model(model, X, y))
                                    desalapp@nd(mesuloneHot"]
                                    #######
                                    ########
                                    Avenage accuracy score: 0.81122222222222
                                    Logistic Regression: 0.81877777777777
In [124]: from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
result= ["LogisticRegression", "OneHot"]
result.extend(evaluate_model(model, X, y))
                                    data.append(result)
                                    \verb|C:\USers\SHEKHAR| anaconda 3 \lib\site-packages \\ | sklearn\linear\_model\_logistic.py: 814: Convergence \\ | convergence \\ 
                                    eWarning:
                                    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/stabl
                                     e/modules/preprocessing.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/modules/linear_model.html#logistic-regression)
                                    \verb|C:\Users\SHEKHAR\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py: 814: Convergence to the convergence of the 
                                     eWarning:
                                    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:
                                                  e/modules/preprocessing.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/modules/linear_model.html#logistic-regression)
                                    C:\Users\SHEKHAR\anaconda3\lib\site-packages\sklearn\linear model\ logistic.py:814: Convergenc
                                     eWarning:
                                    lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
                                    Please also refer to the documentation for alternative solver options:
In [125]: fromhskeardsolve_Dayes.immgostaGaussadanNBs/linear_model.html#logistic-regression (https://scimodelearGausgianNB()/modules/linear_model.html#logistic-regression)
                                    result= ["GaussianNB", "OneHot"]

        CesuserexsHERMHARAanatonmodel(twodele-packages\sklearn\linear_model\_logistic.py:814: Convergenc

                                    dwww.anapgend(result)
                                    ቹቼቹቼቼቹቹቹiled to converge (status=1):
Βኞθρ∏t℥OTAL NO. of ITERATIONS REACHED LIMIT.
                                     ########
                                    e/modules/preprocessing.html)
                                    Please also refer to the documentation for alternative solver options:

#Data Preprocessing WDE Encoding in the processing the processing with the
                                    Handling columns
                                     Results 1.Creating copy of eda_df i.e eda data frame into new data frame for data processing
In [126]: Avecegeedcdfredg_dfocepy(0.819555555555556
```

Handling rows

Convert Categorical columns into numeric using Weight Of Evidence encoding

```
In [127]: ! pip install category_encoders
```

```
Increase the number of iterations (max_iter) or scale the data as shown in:
                                KNNttps://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
                                 Please also refer to the documentation for alternative solver options:
                                fromhskhearmsnäive_bayes.imgøstaGaussidnNBs/linear_model.html#logistic-regression (https://sci
In [125]:
                                modelearGausgianNB()/modules/linear_model.html#logistic-regression)
result= ["GaussianNB", "OneHot"]
                                Cequitre\$H6HHAR\harahuatondad\fuqdelg-packages\sklearn\linear_model\_logistic.py:814: Convergenc
                                 datanangend(result)
                                ######iled to converge (status=1):
                                 BEOP1t$OTAL NO. of ITERATIONS REACHED LIMIT.
                                Increase the number of iterations (max_iter) or scale the data as shown in:
Averagepaccusacyiscomern.0r33477077777777778/preprocessing.html (https://scikit-learn.org/stabl
                                   e/modules/preprocessing.html)
                                Please also refer to the documentation for alternative solver options: #Data Preprocessing WDE Encoding to the control of the 
                                kit-learn.org/stable/modules/linear_model.html#logistic-regression)
                                Handling columns
                                 Results
1. Tealing copy of eda_df i.e eda data frame into new data frame for data processing
```

In [126]: Avecegeed_dfiredy_dfocepy().81955555555555

Handling rows

Convert Categorical columns into numeric using Weight Of Evidence encoding

```
In [127]: ! pip install category encoders
                                      Requirement\ already\ satisfied:\ category\_encoders\ in\ c:\users\shekhar\anaconda3\lib\site-packag
                                      es (2.5.1.post0)
                                      Requirement already satisfied: numpy>=1.14.0 in c:\users\shekhar\appdata\roaming\python\python
                                     39\site-packages (from category_encoders) (1.23.1)
Requirement already satisfied: scikit-learn>=0.20.0 in c:\users\shekhar\anaconda3\lib\site-pac
                                     kages (from category_encoders) (1.0.2)
Requirement already satisfied: scipy>=1.0.0 in c:\users\shekhar\anaconda3\lib\site-packages (f
                                     rom category_encoders) (1.8.1)
Requirement already satisfied: pandas>=1.0.5 in c:\users\shekhar\anaconda3\lib\site-packages
                                       (from category_encoders) (1.4.2)
                                     Requirement already satisfied: patsy>=0.5.1 in c:\users\shekhar\anaconda3\lib\site-packages (f rom category_encoders) (0.5.2)
                                     Requirement already satisfied: statsmodels>=0.9.0 in c:\users\shekhar\anaconda3\lib\site-packa
                                      ges (from category_encoders) (0.13.2)
                                     Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\shekhar\appdata\roaming\pyth on\python39\site-packages (from pandas>=1.0.5->category_encoders) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\shekhar\anaconda3\lib\site-packages (f
                                     rom pandas>=1.0.5->category_encoders) (2021.3)
Requirement already satisfied: six in c:\users\shekhar\appdata\roaming\python\python39\site-packages (from patsy>=0.5.1->category_encoders) (1.16.0)
                                      Requirement already satisfied: joblib>=0.11 in c:\users\shekhar\anaconda3\lib\site-packages (f
                                    rom scikit-learn>=0.20.0->category_encoders) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\shekhar\anaconda3\lib\site-packages (from scikit-learn>=0.20.0->category_encoders) (2.2.0)
Requirement already satisfied: packaging>=21.3 in c:\users\shekhar\appdata\roaming\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\python\pytho
                                     on39\site-packages (from statsmodels>=0.9.0->category_encoders) (21.3) Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\shekhar\appdata\roaming\py
                                      thon\python 39\site-packages (from packaging >= 21.3-> statsmodels >= 0.9.0-> category\_encoders) (3.3-> statsmodels >= 0.9.0-> catego
In [128]: # Drop the Education columns
                                     processed_df.drop('Education', axis=1, inplace=True)
                                      # Work, Country, and Occupation column have '?' values. We will introduce the new category for
# Work, Country, and Occupation column have '?' values.

df['Work'].replace('?','Not Known', inplace=True)

df['Coupation'].replace('?','Not Known', inplace=True)

In [132]:

from sklearn.preprocessing import StandardScaler
In [129]: #calecytn&tondand&calebalang to the Dataframe
                                       from category_encoders.woe import WOEEncoder
categorical_cols = ['Work', 'Marital Status', 'Occupation', 'Gender', 'Country']
In [134]: $\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tilde{\tiiide{\tilde{\tilde{\tilde{\tilde{\tilde{\tiide{\tiide{\tilde{\tiide{\tiide{\tilde{\tilde{\tii
In [135]: #e#edfre ppptpssgdfdftaopy{}lection let's just save the Dependent column
                                     y = one_hot_df['Above 50K']
                                     x = woe_df_scaled_df
In [130]: wef_df.head()
In [136]: from sklearn.decomposition_import_PCA
                                                                                                                                                                                                                                   Gender Occupation Age
                                     pca = PMZAr(k) Marital Status Hours_Week Country Above 50K
                                     0 0.209936 0.948163 13 0.024828 0 0.313434 # Get the principle components for WeF dataframe
                                                                                                                                                                                                                                                                      1.079632
                                                                                                                                                                                                                                                                                                      50
                                     principd254component7370ef = pca.fbt_tro2482form(x) 0 0.313434
                                                                                                                                                                                                                                                               -1.519335
                                                                                                                                                                                                                                                                                                     38
                                       2 -0.120254
                                                                                                                                                 40 0.024828
                                                                                           0.948163
                                                                                                                                                                                                                     0 0.313434 -1.519335 53
In [137]:  pd. \underline{DataEsame}(np.\underline{GymSym}(pca.explained.eyagiance\_ratio_))_{0.888444} 
                                                                                                                                                                                                                                                                     0.941902
                                                                                                                                                                                                                                                                                                     28
Out[137]: 4 -0.120254
                                                                                            0.948163
                                                                                                                                                   40 0.024828
                                                                                                                                                                                                                       0 -0.888444
                                                                                                                                                                                                                                                                     1.079632
 \label{eq:linear_possed_df.to_csv('drive/MyDrive/Inventory\ Project/woe_processed_df.csv')} In \ [131]: \ \# \ p = 299998 \ d_df.to_csv('drive/MyDrive/Inventory\ Project/woe_processed_df.csv')
```

Feature Selection (WOE)

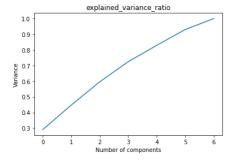
4 0.829041

1 0 447477

5 0.930157

```
In [132]: df['Occupation'].replace('?', 'Not Known', inplace=True) from sklearn.preprocessing import StandardScaler
In [129]: scalptytngtondandscalending to the Dataframe from category_encoders.woe import WOEEncoder
In [133]: WOE_encoder = WOEEncoder()
Above 50K', axis=1)
            categorical_cols = ['Work', 'Marital Status', 'Occupation', 'Gender', 'Country']
In [134]: 해현은 원학보병원리는 원학보통원리는원원로부탁원하는(scaler.fit_transform(scaled_df), columns=scaled_df.columns)
processed_df[column] = WOE_encoder.fit_transform(processed_df[column], processed_df['Above
In [135]: #e#edfre pppt:ssgdfdftropy()lection let's just save the Dependent column
            y = one_hot_df['Above 50K']
            x = woe_df_scaled_df
In [130]: wef_df.head()
In [136]: from sklearn.
            from sklearn.decomposition import PCA
Out[130]:
            pca = PMZAr(k) Marital Status Hours_Week Country Above 50K
            # Get the principle components for WeF dataframe
                                                                       0 0.313434
                                                                                      1.079632
                                                                                                  50
            principde54componentx370ef = pca.fipt_promasform(x)
                                                                      0 0.313434
                                                                                      -1.519335
                           0.948163
             2 -0.120254
                                                 40 0.024828
                                                                       0 0.313434
                                                                                      -1.519335
In [137]: pd.DataFrame(np.cymsum(pca.explained_variance_ratio_)).888444
                                                                                      0.941902 28
                                                40 0.024828
                                                                       0 -0.888444
Out[137]: 4 -0.120254
                              0.948163
                                                                                      1.079632 37
In [131]: #0 p029995d_df.to_csv('drive/MyDrive/Inventory Project/woe_processed_df.csv')
            1 0.447477
              0 506107
            Éeature Selection (WOE)
             4 0.829041
             5 0.930157
             6 1.000000
```

```
In [138]: plt.figure()
   plt.plot(np.cumsum(pca.explained_variance_ratio_))
   plt.xlabel('Number of components')
   plt.ylabel('Variance')
   plt.title('explained_variance_ratio')
   plt.show()
```



We can see that around 95% of variance is being explained by 5 components. So instead of giving all columns as input in our algorithm let's use these principle components instead

```
In [139]: pca = PCA(n_components=5)
              pca\_woe\_df = pca.fit\_transform(x)
              column_name = [f'PC-{i}' for i in range(1, 6)]
              principal_woe_df = pd.DataFrame(pca_woe_df, columns=column_name)
prmartpaklaaendmekeaat)as sm
from sklearn.model_selection import KFold
In [140]:
Out[139]:
              # Function to evaluate mades
              def evaluate model(model, x, y):

0 1022950y 0.5027e7 =1.834604 0.483262 -2.233079
              1 -@6643631-08.6696695 6183460 -0.623830 -0.104710 recall_scores = [] 2 045-45067e3-586618 -0.702159 -0.355450 0.091552
              3 -0.112656 0.537810 1.162723 1.649826 0.374699
kf = KFold(n_splits=10, random_state=0, shuffle=True)
4 0623505ain_588565, 0685692ndex5747kf.841847(x):
                         #setting up the data
              ##Data Modeling n, v_test = x.values[train_index], x.values[test_index] ##Data Modeling n, v_test = y.values[train_index], y.values[test_index]
                         #Training model
                        model.fit(x_train,y_train)
                         #Evaluating model
                        y_pred = model.predict(x_test)
                         accuracy_scores.append(sm.accuracy_score(y_test,y_pred))
                         precision scores.append(sm.precision score(y test,y pred))
                         recall_scores.append(sm.recall_score(y_test,y_pred))
                         f1_scores.append(sm.f1_score(y_test, y_pred))
                   #displaying average results
                   print("######")
                   print("#######\n")
                   print("Average accuracy score: ", (sum(accuracy scores)/len(accuracy scores)))
                   results = [(sum(accuracy_scores))]
```

```
principal_woe_df = pd.DataFrame(pca_woe_df, columns=column_name)
In [140]: jmpnctpsklmaendmenteigs)as sm
             from sklearn.model_selection import KFold
Out[139]:
             # FunctRom to evad-gate madel
            def evaluate model(model, x, y):
0 1022950y 0.5007e3 =1.334604 0.483262 -2.233079
             3 -0.112656 0.537810 1.162723 1.649826 0.374699
kf = KFold(n_splits=10, random_state=0, shuffle=True)
4 ዓርቶ3505ain_596565, ዓርፅ5654nd&557477kf.8ታ18ቲቪኒ):
            #setting up the data
x train, x_test = x.values[train_index], x.values[test_index]
##Data Modeling n, y_test = y.values[train_index], y.values[test_index]
                      model.fit(x train,y train)
                      #Evaluating model
                      y_pred = model.predict(x_test)
                      accuracy scores.append(sm.accuracy score(y test,y pred))
                      precision_scores.append(sm.precision_score(y_test,y_pred))
recall_scores.append(sm.recall_score(y_test,y_pred))
                      f1_scores.append(sm.f1_score(y_test, y_pred))
                 #displaying average results
print("#######")
                 print("Results")
print("#######\n")
                 print("Average accuracy score: ", (sum(accuracy_scores))len(accuracy_scores)))
                 results = [(sum(accuracy_scores))]
            ###Random Forest
In [141]: import numpy as np # linear algebra
            import pandas as p# data processing, CSV file I/O (e.g. pd.read_csv) from sklearn.preprocessing import LabelEncoder
            import matplotlib.pyplot as plt
import seaborn as sns
            %matplotlib inline
            import lightgbm as lgb
In [142]: y=wef_df['Above 50K']
            X= principal_woe_df
In [143]: from sklearn.ensemble import RandomForestClassifier
    model = RandomForestClassifier()
            result= ["Random Forest", "WOE"]
result.extend(evaluate_model(model, X, y))
            data.append(result)
            #######
            Results
            Average accuracy score: 0.796333333333333333
            ###Decision Tree
In [144]: from sklearn.tree import DecisionTreeClassifier
            model = DecisionTreeClassifier()
model = XGBClassifier()
            result.extend(evaluate_model(model,X, y))
             Results
            #######
            <u>A¥ዋዋа89</u>#accuracy score: 0.76288888888888888
            ###Adaboost
In [145]: from sklearn.svm import SVC
            from sklears, ensemble import AdaBoostClassifier model = AdaBoostClassifier(random_state=0) result= ["Adaboost", WOE"]
            result:extend(evaluate_model(model; %; y))
data:append(result)
            #######
Results
             Average accuracy score: 0:3932222222222221
            ##gisticsRegression
In [148]: from sklearn.linear model import LogisticRegression
            model = LogisticRegression()
result= ["LogisticRegression", "WOE"]
            result.extend(evaluate model(model, X, y))
```

```
In [146]: FFSMlkgeKStdfmKaltakGeFodelfmodel, X, y))
#eSGItLireXtBEGGevalumGe_model(model, X, y))
data.append(result)
              model = XGBClassifier()
              result.extend(evaluate_model(model,X, y))
問題報事的中的(result)
Results
              #######
              Results
              Average accuracy score: 0.7971111111111111
              SVM
              ###Adaboost
In [145]: from sklearn.svm import SVC
             #Bogm:sklearn,ensemble import AdaBoostClassifier mgdel.= AdaBoostClassifier(random_state=0) result= ["AdaboostC, woe"] result:extend(evaluate_msdel(msdel; %; y)) data:append(result)
In [147]:
              Average accuracy score: 0:393222222222221
              ##gistic Regression
In [148]: from sklearn.linear_model import LogisticRegression
             model = LogisticRegression()
result= ["LogisticRegression", "WOE"]
result.extend(evaluate_model(model, X, y))
              data.append(result)
              Results
              ########
              Average accuracy score: 0.7936666666666667
              KNN
In [149]: from sklearn.naive_bayes import GaussianNB
              model = GaussianNB()
result= ["GaussianNB", "WOE"]
              result.extend(evaluate_model(model, X, y))
              data.append(result)
              ########
              Results
              #######
              Average accuracy score: 0.79477777777778
              Saving Results for Income Prediction
In [150]: results_df = pd.DataFrame(data,columns=columns)
    results_df = results_df[['Algorithm', 'Encoding', 'Accuracy']]
    results_df.sort_values(by=['Accuracy', 'Encoding'], ascending=False)
Out[150]:
                           Algorithm Encoding Accuracy
               5 LogisticRegression OneHot 0.819556
                2
                               SVM OneHot 0.818778
In [151]: sns.barplot(data_mresultsoeff,ownicate), y='Algorithm', hue='Encoding') plt.legend(loc='lower left')

4 Adaboost OneHot 0.811222
Out[151]: <matplotlib_legend.Legend_at_0215e5742b0a0>
               11
                     Adaboo
Random Forest -
XGBoo
               10
                      Decision tree
Random For
                7
                         Gaussyant
               60
                       RandloftoFid
                   LogisticRegress
                1 Decision to 
LogisticRegression
                        Decision tr
                                                             0.4 0.5
Accuracy
                                        0.1
                                               0.2
                                                     0.3
In [152]: results_df.to_csv("income_prediction_results.csv")
```

```
In [151]: sng.barplot(dagagresultwoff, 0.%150ccuracy', y='Algorithm', hue='Encoding')
plt.legend(loc='lower left')
4 Adaboost OneHol 0.811222

Out[151]: <matplotlib_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legend_Legen
```