MANOJ KUMAR - 2048015

Requirement

For the given anonymous dataset of size 199x35 perform the following task:

1. Exploratory Data analysis to study the nature of the data and to decide whether to follow a parametric approach or non parametric approach for predicting the target.
2. Preprocessing
3. Dimensionality reduction
4. Model building
5. Model Evaluation

NOTE: Register Number 1 to 20 will perform prediction on column named predictLabel2(continuous value)

In [1]:
```
#Importing libraries
import numpy as np
import pandas as pd

#Importing the visualisation libraries
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [33]:
```
#Reading the data
MainDataset = pd.read_csv('AnonymousDataset.csv')
best_df = MainDataset
data = MainDataset
```

Perform Exploratory data analysis

In [3]:
```
MainDataset.head(3)
```

Out[3]:

	col1	classLabel	col3	col4	col5	col6	col7	col8	col9	col10	...	col26	col27	col28	col29	col30	col31	col32	col33
0	119513	0	31	18.02	27.60	117.5	1013.0	0.09489	0.1036	0.1086	...	139.7	1436.0	0.1195	0.1926	0.3140	0.1170	0.2677	0.08113
1	8423	0	61	17.99	10.38	122.8	1001.0	0.11840	0.2776	0.3001	...	184.6	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890
2	842517	0	116	21.37	17.44	137.5	1373.0	0.08836	0.1189	0.1255	...	159.1	1949.0	0.1188	0.3449	0.3414	0.2032	0.4334	0.09067

3 rows × 35 columns

Regression predictive modeling problem.

In [4]:
```
print(f"Totally AnonymousDataset contains, {MainDataset.shape[1]} columns and {MainDataset.shape[0]} Rows")
```

```
Totally AnonymousDataset contains, 35 columns and 198 Rows
```

In [5]:
```
MainDataset.head().T
```

Out[5]:

	0	1	2	3	4
col1	119513	8423	842517	843483	843584
classLabel	0	0	0	0	1
col3	31	61	116	123	27
col4	18.02	17.99	21.37	11.42	20.29
col5	27.6	10.38	17.44	20.38	14.34
col6	117.5	122.8	137.5	77.58	135.1
col7	1013	1001	1373	386.1	1297
col8	0.09489	0.1184	0.08836	0.1425	0.1003
col9	0.1036	0.2776	0.1189	0.2839	0.1328
col10	0.1086	0.3001	0.1255	0.2414	0.198
col11	0.07055	0.1471	0.0818	0.1052	0.1043
col12	0.1865	0.2419	0.2333	0.2597	0.1809
col13	0.06333	0.07871	0.0601	0.09744	0.05883
col14	0.6249	1.095	0.5854	0.4956	0.7572
col15	1.89	0.9053	0.6105	1.156	0.7813
col16	3.972	8.589	3.928	3.445	5.438
col17	71.55	153.4	82.15	27.23	94.44
col18	0.004433	0.006399	0.006167	0.00911	0.01149
col19	0.01421	0.04904	0.03449	0.07458	0.02461
col20	0.03233	0.05373	0.033	0.05661	0.05688
col21	0.009854	0.01587	0.01805	0.01867	0.01885
col22	0.01694	0.03003	0.03094	0.05963	0.01756
col23	0.003495	0.006193	0.005039	0.009208	0.005115
col24	21.63	25.38	24.9	14.91	22.54
col25	37.08	17.33	20.98	26.5	16.67
col26	139.7	184.6	159.1	98.87	152.2
col27	1436	2019	1949	567.7	1575
col28	0.1195	0.1622	0.1188	0.2098	0.1374
col29	0.1926	0.6656	0.3449	0.8663	0.205
col30	0.314	0.7119	0.3414	0.6869	0.4
col31	0.117	0.2654	0.2032	0.2575	0.1625
col32	0.2677	0.4601	0.4334	0.6638	0.2364
col33	0.08113	0.1189	0.09067	0.173	0.07678
PredictLabel1	5	3	2.5	2	3.5
PredictLabel2	5	2	0	0	0

In [6]:
```
MainDataset.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
col1	198.0	1.990469e+06	2.889025e+06	8423.000000	855745.250000	886339.000000	927995.750000	9.411300e+06
classLabel	198.0	2.373737e-01	4.265517e-01	0.000000	0.000000	0.000000	0.000000	1.000000e+00
col3	198.0	4.673232e+01	3.446287e+01	1.000000	14.000000	39.500000	72.750000	1.250000e+02
col4	198.0	1.741232e+01	3.161676e+00	10.950000	15.052500	17.290000	19.580000	2.722000e+01
col5	198.0	2.227601e+01	4.298290e+00	10.380000	19.412500	21.750000	24.655000	3.928000e+01
col6	198.0	1.148566e+02	2.138340e+01	71.900000	98.160000	113.700000	129.650000	1.821000e+02
col7	198.0	9.700409e+02	3.521492e+02	361.600000	702.525000	929.100000	1193.500000	2.250000e+03
col8	198.0	1.026814e-01	1.252243e-02	0.074970	0.093900	0.101900	0.110975	1.447000e-01
col9	198.0	1.426478e-01	4.989760e-02	0.046050	0.110200	0.131750	0.172200	3.114000e-01
col10	198.0	1.562428e-01	7.057226e-02	0.023980	0.106850	0.151350	0.200500	4.268000e-01
col11	198.0	8.677561e-02	3.387663e-02	0.020310	0.063670	0.086075	0.103925	2.012000e-01
col12	198.0	1.927540e-01	2.743689e-02	0.130800	0.174075	0.189350	0.209325	3.040000e-01
col13	198.0	6.270551e-02	7.239530e-03	0.050250	0.056718	0.061715	0.066715	9.744000e-02
col14	198.0	6.033465e-01	3.101122e-01	0.193800	0.388200	0.533250	0.750900	1.819000e+00
col15	198.0	1.264450e+00	5.264669e-01	0.362100	0.921300	1.168500	1.463250	3.503000e+00
col16	198.0	4.255394e+00	2.194128e+00	1.153000	2.742500	3.767000	5.212750	1.328000e+01
col17	198.0	7.022874e+01	4.798225e+01	13.990000	35.365000	58.455000	92.477500	3.160000e+02
col18	198.0	6.761864e-03	2.974270e-03	0.002667	0.005001	0.006193	0.007973	3.113000e-02
col19	198.0	3.119929e-02	1.761293e-02	0.007347	0.019803	0.027880	0.038335	1.354000e-01
col20	198.0	4.074980e-02	2.086872e-02	0.010940	0.026810	0.036910	0.048970	1.438000e-01
col21	198.0	1.509925e-02	5.504267e-03	0.005174	0.011422	0.014175	0.017665	3.927000e-02
col22	198.0	2.055486e-02	9.578243e-03	0.007882	0.014795	0.017905	0.022880	6.041000e-02
col23	198.0	3.986904e-03	1.937845e-03	0.001087	0.002748	0.003719	0.004630	1.256000e-02
col24	198.0	2.102182e+01	4.242997e+00	12.840000	17.632500	20.525000	23.730000	3.513000e+01
col25	198.0	3.013909e+01	6.017777e+00	16.670000	26.210000	30.135000	33.555000	4.954000e+01
col26	198.0	1.403478e+02	2.889228e+01	85.100000	118.075000	136.500000	159.875000	2.322000e+02
col27	198.0	1.404959e+03	5.860070e+02	508.100000	947.275000	1295.000000	1694.250000	3.903000e+03
col28	198.0	1.439208e-01	2.200396e-02	0.081910	0.129325	0.141850	0.154875	2.226000e-01
col29	198.0	3.651018e-01	1.639650e-01	0.051310	0.248700	0.351300	0.423675	1.058000e+00
col30	198.0	4.366853e-01	1.736245e-01	0.023980	0.322150	0.402350	0.541050	1.170000e+00
col31	198.0	1.787775e-01	4.518052e-02	0.028990	0.152650	0.179250	0.207125	2.903000e-01
col32	198.0	3.234040e-01	7.516089e-02	0.156500	0.275950	0.310300	0.358800	6.638000e-01
col33	198.0	9.082813e-02	2.117197e-02	0.055040	0.076578	0.086890	0.101375	2.075000e-01
PredictLabel1	198.0	2.847475e+00	1.937964e+00	0.400000	1.500000	2.500000	3.500000	1.000000e+01

In [7]:
```
MainDataset.isnull().sum().sort_values(ascending=False)
```

Out[7]:
```
PredictLabel2    0
col9             0
col15            0
col14            0
col13            0
col12            0
col11            0
col10            0
col8             0
col17            0
col7             0
col6             0
col5             0
col4             0
col3             0
classLabel       0
col16            0
col18            0
PredictLabel1    0
col27            0
col33            0
col32            0
col31            0
col30            0
col29            0
col28            0
col26            0
col19            0
col25            0
col24            0
col23            0
col22            0
col21            0
col20            0
col1             0
dtype: int64
```

In [8]:
```
for i in MainDataset.columns:
    print(f'{i} \t \t \t : \t {MainDataset[i].nunique()} values')
```

```
col1                     :       198 values
classLabel                       :        2 values
col3                     :       95 values
col4                     :       177 values
col5                     :       193 values
col6                     :       181 values
col7                     :       192 values
col8                     :       179 values
col9                     :       192 values
col10                    :       196 values
col11                    :       189 values
col12                    :       175 values
col13                    :       194 values
col14                    :       196 values
col15                    :       191 values
col16                    :       192 values
col17                    :       196 values
col18                    :       196 values
col19                    :       193 values
col20                    :       192 values
col21                    :       187 values
col22                    :       189 values
col23                    :       195 values
col24                    :       182 values
col25                    :       187 values
col26                    :       183 values
col27                    :       191 values
col28                    :       172 values
col29                    :       191 values
col30                    :       197 values
col31                    :       185 values
col32                    :       192 values
col33                    :       189 values
PredictLabel1                    :        39 values
PredictLabel2                    :        23 values
```

In [9]:
```
numerical_features = []
categorical_features = []

for i in MainDataset.columns:
    if MainDataset[i].nunique()>7:
        numerical_features.append(i)
    else:
        categorical_features.append(i)


print(len(numerical_features))
print(len(categorical_features))
```

```
34
1
```

In [10]:
```
# Numerical features:
print("Numerical features : ",numerical_features)

# Categorical features:
print("\n Categorical features : ",categorical_features)
```

```
Numerical features :  ['col1', 'col3', 'col4', 'col5', 'col6', 'col7', 'col8', 'col9', 'col10', 'col11', 'col
12', 'col13', 'col14', 'col15', 'col16', 'col17', 'col18', 'col19', 'col20', 'col21', 'col22', 'col23', 'col2
4', 'col25', 'col26', 'col27', 'col28', 'col29', 'col30', 'col31', 'col32', 'col33', 'PredictLabel1', 'Predic
tLabel2']

 Categorical features :  ['classLabel']
```

In [11]:
```
# checking for unique values in categorical features:

for feats in categorical_features:
    print(f'{feats} has {MainDataset[feats].unique()} categories.\n')
```

```
classLabel has [0 1] categories.
```

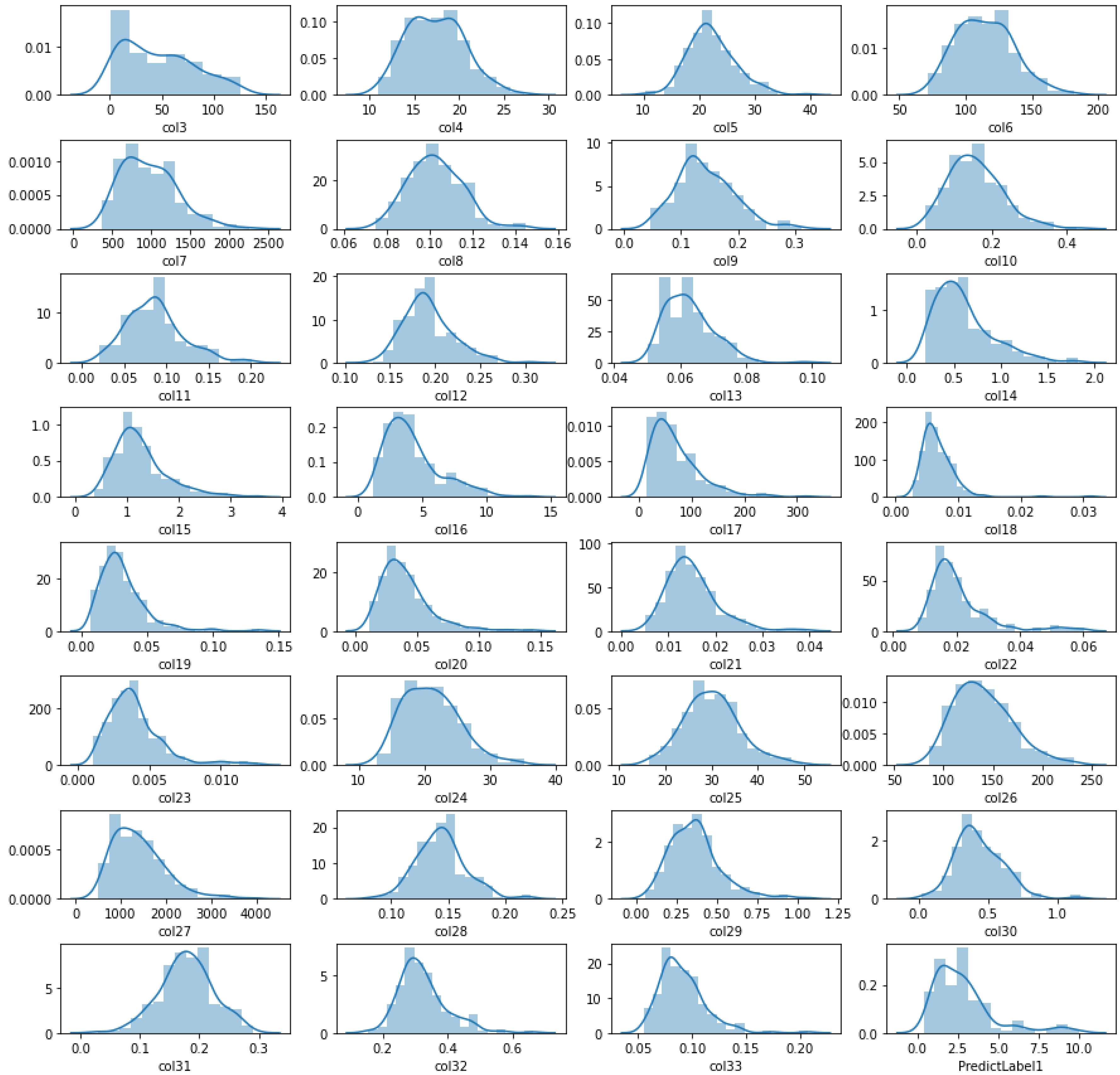
In [12]:
```
numerical_features.remove('col1')
```

In [13]:
```python
# Checking distribution of the numerical features:

fig, axes = plt.subplots(nrows=8, ncols=4, figsize=(15,15))
fig.subplots_adjust(hspace=0.5)
fig.suptitle('Distributions of numerical Features')


for ax, feats in zip(axes.flatten(), numerical_features):
    sns.distplot(a=MainDataset[feats], ax=ax)
```

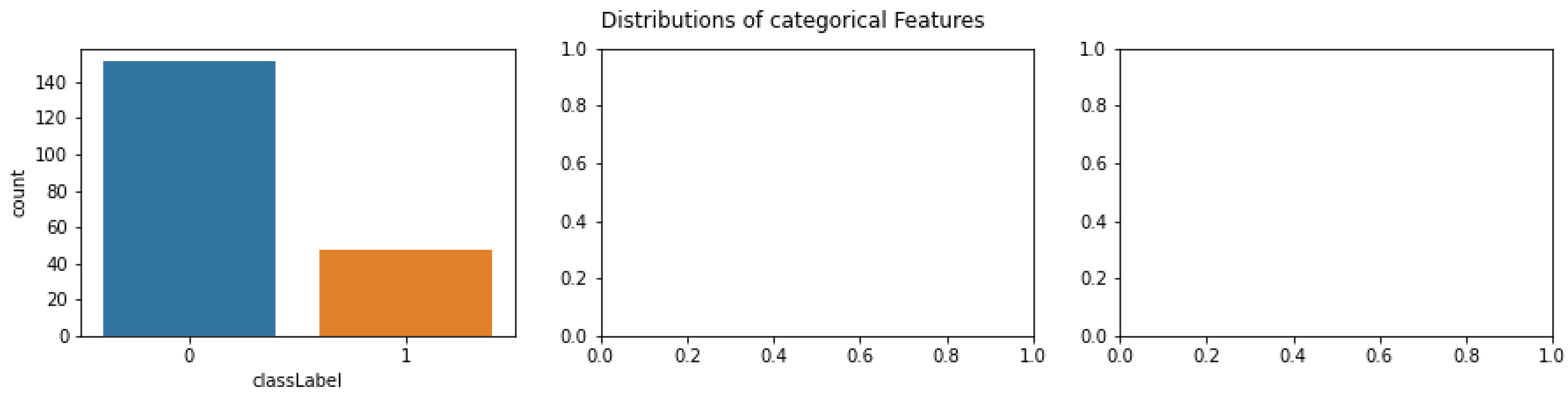
Distributions of numerical Features



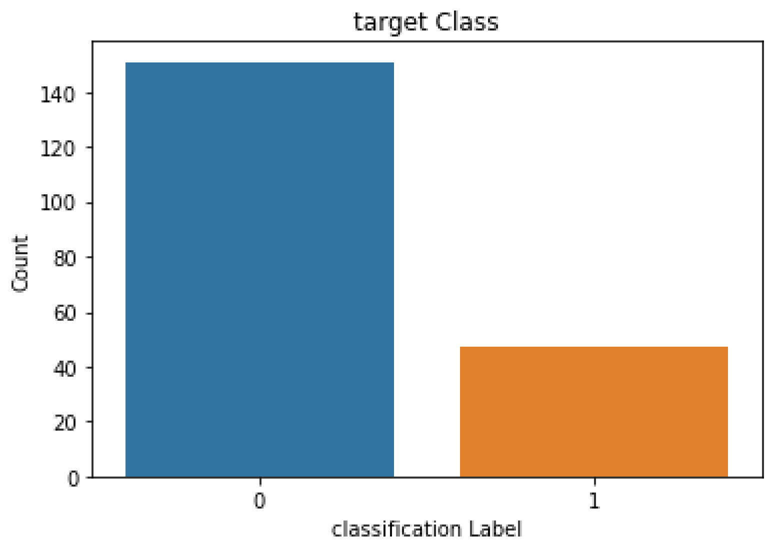
In [14]:
```python
# Checking the label distribution for categorical data:

fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15,3))
fig.subplots_adjust(hspace=0.5)
fig.suptitle('Distributions of categorical Features')


for ax, feats in zip(axes.flatten(), categorical_features):
    sns.countplot(MainDataset[feats], ax=ax)
```



In [15]:
```python
sns.countplot(x='classLabel',data=MainDataset)
plt.xlabel("classification Label")
plt.ylabel("Count")
plt.title("target Class")
plt.show()
```

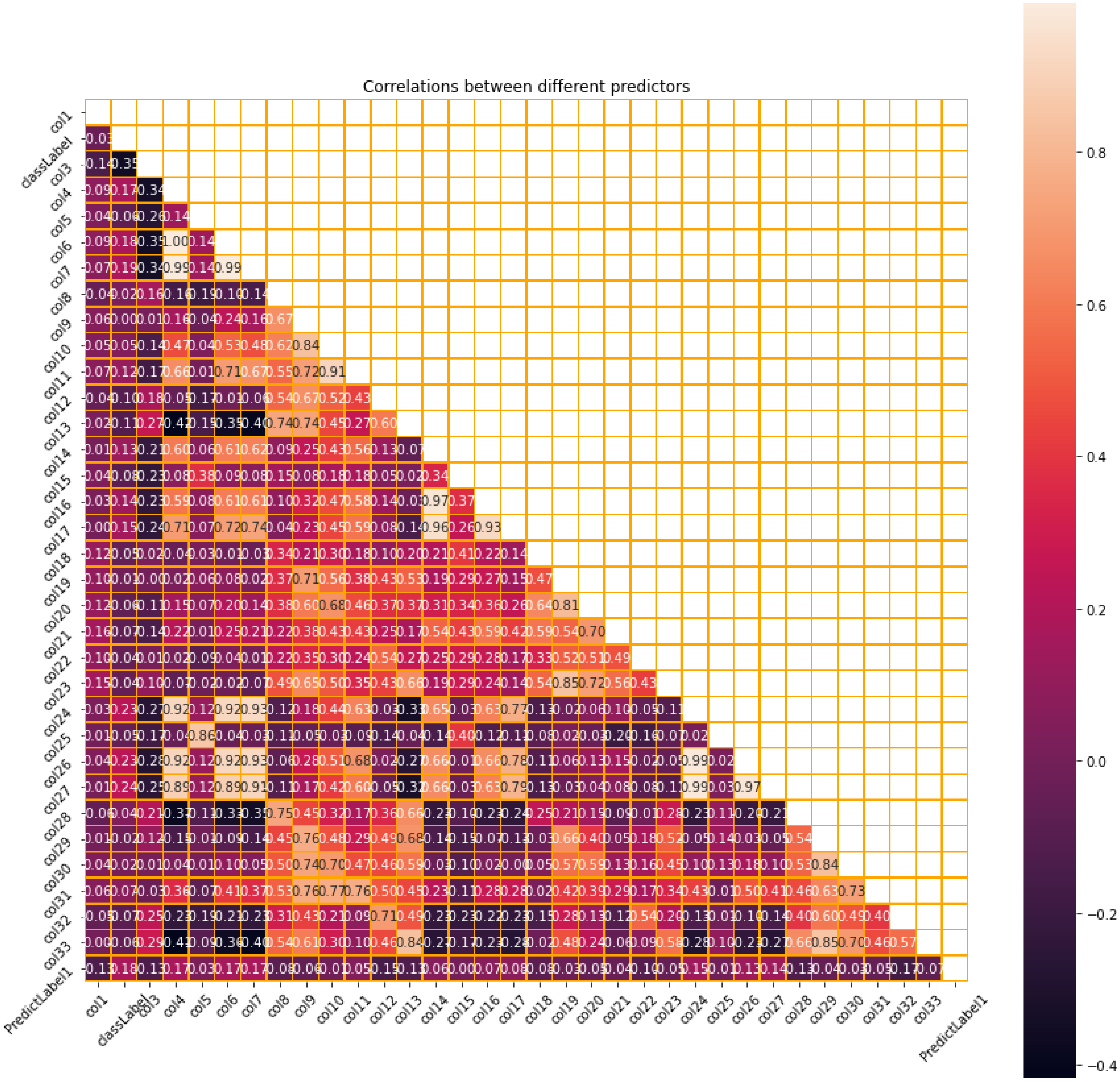


Data cleaning & Pre-processing

In [17]:
```python
corr_df = MainDataset.corr()

f,ax=plt.subplots(figsize=(15,15))
mask = np.zeros_like(corr_df)
mask[np.triu_indices_from(mask)] = True

sns.heatmap(corr_df,annot=True,fmt=".2f",ax=ax,linewidths=0.5,linecolor="orange", mask = mask, square=True)
plt.xticks(rotation=45)
plt.yticks(rotation=45)
plt.title('Correlations between different predictors')
plt.show()
```



In [18]:
```python
df = MainDataset
```

In [19]:
```python
corr_df = df.corr()
corr_df
cm = sns.light_palette("brown", as_cmap=True)
corr_df.style.background_gradient(cmap=cm)
```

Out[19]:

	col1	classLabel	col3	col4	col5	col6	col7	col8	col9	col10	col11	col12	
col1	1.000000	-0.031466	-0.135299	0.087392	0.037650	0.088027	0.070117	-0.039803	0.059505	0.051946	0.074368	-0.043264	0.0
classLabel	-0.031466	1.000000	-0.351326	0.174124	-0.064295	0.176486	0.189893	0.020778	0.000798	0.054893	0.118224	-0.099777	-0.1
col3	-0.135299	-0.351326	1.000000	-0.344722	-0.264671	-0.346080	-0.344031	0.164793	0.010000	-0.139475	-0.171841	0.177311	0.2
col4	0.087392	0.174124	-0.344722	1.000000	0.143456	0.995933	0.992855	-0.158239	0.159017	0.469518	0.664010	-0.051610	-0.4
col5	0.037650	-0.064295	-0.264671	0.143456	1.000000	0.142033	0.140440	-0.192262	-0.039803	0.037165	0.006687	-0.165166	-0.1
col6	0.088027	0.176486	-0.346080	0.995933	0.142033	1.000000	0.990699	-0.102912	0.236721	0.533194	0.712766	-0.006512	-0.3
col7	0.070117	0.189893	-0.344031	0.992855	0.140440	0.990699	1.000000	-0.141470	0.163176	0.475862	0.667530	-0.060785	-0.3
col8	-0.039803	0.020778	0.164793	-0.158239	-0.192262	-0.102912	-0.141470	1.000000	0.666559	0.623867	0.545734	0.540761	0.7
col9	0.059505	0.000798	0.010000	0.159017	-0.039803	0.236721	0.163176	0.666559	1.000000	0.836015	0.716438	0.666822	0.7
col10	0.051946	0.054893	-0.139475	0.469518	0.037165	0.533194	0.475862	0.623867	0.836015	1.000000	0.909990	0.524861	0.4
col11	0.074368	0.118224	-0.171841	0.664010	0.006687	0.712766	0.667530	0.545734	0.716438	0.909990	1.000000	0.429968	0.2
col12	-0.043264	-0.099777	0.177311	-0.051610	-0.165166	-0.006512	-0.060785	0.540761	0.666822	0.524861	0.429968	1.000000	0.6
col13	0.024509	-0.112352	0.269992	-0.416674	-0.145572	-0.353560	-0.397733	0.744890	0.735474	0.449928	0.268210	0.604104	1.0
col14	0.012313	0.132512	-0.214543	0.602035	0.059168	0.612708	0.623019	0.094728	0.251568	0.427031	0.555034	0.130985	-0.0
col15	0.037530	-0.076212	-0.230477	0.079693	0.382533	0.092256	0.084288	0.153848	0.082994	0.181984	0.179486	0.049953	0.0
col16	0.027710	0.141633	-0.231621	0.588927	0.075025	0.609964	0.609887	0.099518	0.318684	0.468426	0.580562	0.143176	-0.0
col17	0.004872	0.151826	-0.244159	0.710586	0.068517	0.718581	0.740830	0.037955	0.233326	0.449059	0.586508	0.079773	-0.1
col18	0.124037	-0.052213	0.019775	-0.036419	0.027119	-0.011788	-0.032969	0.344678	0.212552	0.297014	0.177402	0.104636	0.1
col19	0.100940	-0.009537	-0.002386	0.023647	0.063988	0.080725	0.020395	0.372393	0.714122	0.564196	0.376339	0.426781	0.5
col20	0.119171	-0.060379	-0.108648	0.154254	0.071920	0.202027	0.144443	0.375011	0.599020	0.676804	0.463710	0.374679	0.3
col21	0.158519	-0.065570	-0.140754	0.224771	0.010470	0.254473	0.213582	0.223510	0.384747	0.434928	0.426638	0.251257	0.1
col22	0.100087	-0.044325	0.011156	0.019146	-0.094843	0.038613	0.009121	0.223723	0.350009	0.304383	0.238520	0.541034	0.2
col23	0.152869	-0.042751	0.099203	-0.072618	-0.020673	-0.019514	-0.071906	0.486112	0.648248	0.504069	0.350050	0.432752	0.6
col24	0.031900	0.233225	-0.265115	0.924183	0.123028	0.921552	0.932381	-0.115092	0.183277	0.437961	0.630309	-0.030809	-0.3
col25	0.007285	-0.051134	-0.171125	-0.039439	0.862050	-0.039728	-0.032122	-0.106172	-0.047665	-0.032081	-0.094163	-0.137598	-0.0
col26	0.043099	0.231998	-0.280596	0.915053	0.123674	0.923659	0.925335	-0.064664	0.276994	0.514336	0.682749	0.019708	-0.2
col27	0.008451	0.235310	-0.253930	0.891489	0.117467	0.889344	0.914166	-0.106691	0.168275	0.421021	0.604029	-0.050522	-0.3
col28	-0.057241	0.038520	0.212769	-0.372894	-0.113308	-0.331667	-0.345111	0.745400	0.452067	0.319247	0.174917	0.355244	0.6
col29	0.005048	-0.020067	0.120516	-0.150712	-0.006467	-0.092041	-0.141358	0.447849	0.764824	0.483300	0.286599	0.488231	0.6
col30	0.043288	0.017621	0.009546	0.038952	0.013635	0.096790	0.046641	0.499438	0.743333	0.702673	0.471429	0.458280	0.5
col31	0.059702	0.074345	-0.026541	0.357869	-0.069921	0.410000	0.365026	0.531015	0.761044	0.767571	0.764803	0.501957	0.4
col32	-0.053920	-0.074731	0.247678	-0.232142	-0.186850	-0.206949	-0.234294	0.308964	0.429953	0.212976	0.089804	0.705076	0.4
col33	0.003154	-0.055170	0.288715	-0.414340	-0.085847	-0.364022	-0.395026	0.535751	0.611315	0.302868	0.101327	0.458548	0.8
PredictLabel1	-0.132809	0.177273	-0.133355	0.172102	0.027073	0.166489	0.174491	-0.084376	-0.060199	-0.010244	0.050040	-0.151551	-0.1

Perform Feature Selection Techniques

Numerical Input, Numerical Output

This is a regression predictive modeling problem with numerical input variables. The most common techniques are to use a correlation coefficient, such as Pearson's for a linear correlation, or rank-based methods for a nonlinear correlation.

1. Pearson's correlation coefficient (linear).
2. Spearman's rank coefficient (nonlinear)

In [20]:
```python
from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.metrics import roc_auc_score

from mlxtend.feature_selection import ExhaustiveFeatureSelector as EFS

import warnings
warnings.filterwarnings('ignore')
```

In [21]:
```python
# Encoding categorical variables into numbers
numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
numerical_vars = list(data.select_dtypes(include=numerics).columns)
data = data[numerical_vars]
data.shape
```

Out[21]: (198, 34)

In [23]:
```python
data
```

Out[23]:

Label	col3	col4	col5	col6	col7	col8	col9	col10	...	col25	col26	col27	col28	col29	col30	col31	col32	col33	Pred
0	31	18.02	27.60	117.50	1013.0	0.09489	0.10360	0.10860	...	37.08	139.70	1436.0	0.11950	0.1926	0.3140	0.11700	0.2677	0.08113	
0	61	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	...	17.33	184.60	2019.0	0.16220	0.6656	0.7119	0.26540	0.4601	0.11890	
0	116	21.37	17.44	137.50	1373.0	0.08836	0.11890	0.12550	...	20.98	159.10	1949.0	0.11880	0.3449	0.3414	0.20320	0.4334	0.09067	
0	123	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	...	26.50	98.87	567.7	0.20980	0.8663	0.6869	0.25750	0.6638	0.17300	
1	27	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	...	16.67	152.20	1575.0	0.13740	0.2050	0.4000	0.16250	0.2364	0.07678	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
0	10	22.52	21.92	146.90	1597.0	0.07592	0.09162	0.06862	...	24.81	162.10	1902.0	0.08191	0.1319	0.1056	0.09378	0.2061	0.05788	
0	8	15.44	31.18	101.00	740.4	0.09399	0.10620	0.13750	...	41.48	112.60	929.0	0.12720	0.2362	0.2975	0.12860	0.2914	0.08024	
0	12	17.17	29.19	110.00	915.3	0.08952	0.06655	0.06583	...	36.66	132.50	1295.0	0.12610	0.1572	0.2141	0.09520	0.3362	0.06033	
1	3	21.42	22.84	145.00	1440.0	0.10700	0.19390	0.23800	...	27.98	198.30	2375.0	0.14980	0.4379	0.5411	0.22150	0.2832	0.08981	
0	6	16.70	28.13	110.30	885.4	0.08896	0.11310	0.10120	...	34.92	128.80	1213.0	0.13300	0.2808	0.3455	0.13170	0.3035	0.08036	

nns

In [24]:
```python
# separate train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    data.drop(labels=["col1"], axis=1),
    data['PredictLabel1'],
    test_size=0.3,
    random_state=0)

X_train.shape, X_test.shape
```

Out[24]: ((138, 33), (60, 33))

In [25]:
```python

# find and remove correlated features
# in order to reduce the feature space a bit
# so that the algorithm takes shorter

def correlation(dataset, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff value
                colname = corr_matrix.columns[i]  # getting the name of column
                col_corr.add(colname)
    return col_corr

corr_features = correlation(X_train, 0.8)
print('correlated features: ', len(set(corr_features)) )
```

correlated features: 14

In [26]:
```python

# removed correlated  features
X_train.drop(labels=corr_features, axis=1, inplace=True)
X_test.drop(labels=corr_features, axis=1, inplace=True)

X_train.shape, X_test.shape
```

Out[26]: ((138, 19), (60, 19))

In [27]:
```python
X_train.columns[0:10]
```

Out[27]:
```
Index(['classLabel', 'col3', 'col4', 'col5', 'col8', 'col9', 'col12', 'col13',
       'col14', 'col15'],
      dtype='object')
```

In [28]:
```python
# exhaustive feature selection
# Using 10 features with ROC_AUC Scoring

efs1 = EFS(RandomForestClassifier(n_jobs=4, random_state=0),
           min_features=1,
           max_features=4,
           scoring='roc_auc',
           print_progress=True,
           cv=2)
```

In [29]:
```python
def run_randomForests(X_train, X_test, y_train, y_test):
    rf = RandomForestClassifier(n_estimators=200, random_state=39, max_depth=4)
    rf.fit(X_train, y_train)
    print('Train set')
    pred = rf.predict_proba(X_train)
    print('Random Forests roc-auc: {}'.format(roc_auc_score(y_train, pred[:,1])))
    print('Test set')
    pred = rf.predict_proba(X_test)
    print('Random Forests roc-auc: {}'.format(roc_auc_score(y_test, pred[:,1])))
```

In [35]:
```python
X_train
```

Out[35]:

	classLabel	col3	col4	col5	col8	col9	col12	col13	col14	col15	col18	col19	col21	col22	col28	col29	col31	
139	0	13	19.55	23.21	0.10100	0.1318	0.1989	0.05884	0.6107	2.8360	0.011240	0.04097	0.03441	0.02768	0.1251	0.2414	0.1825	0
80	0	86	14.22	23.12	0.10750	0.2413	0.2384	0.07542	0.2860	2.1100	0.007970	0.13540	0.01666	0.05113	0.1533	0.9327	0.1772	0
19	0	116	17.14	16.40	0.11860	0.2276	0.3040	0.07413	1.0460	0.9760	0.008029	0.03799	0.02397	0.02308	0.1545	0.3949	0.2550	0
159	1	11	20.59	21.24	0.10850	0.1644	0.1848	0.06222	0.5904	1.2160	0.006666	0.02791	0.01479	0.01117	0.1464	0.3597	0.2113	0
90	0	74	17.42	25.56	0.10060	0.1146	0.1308	0.05866	0.5296	1.6670	0.031130	0.08555	0.03927	0.02175	0.1243	0.1793	0.1099	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
67	1	44	17.68	20.74	0.11150	0.1665	0.1971	0.06166	0.8113	1.4000	0.009037	0.04954	0.01841	0.01778	0.1418	0.3498	0.1515	0
192	0	3	14.72	25.26	0.11740	0.2112	0.2079	0.07496	0.3405	1.1580	0.004957	0.04553	0.01597	0.02539	0.1464	0.5352	0.1974	0
117	0	17	19.71	19.06	0.10180	0.1352	0.1895	0.05863	0.4352	1.0490	0.004996	0.02395	0.01117	0.02266	0.1411	0.3993	0.1925	0
47	0	97	19.55	15.49	0.10790	0.1747	0.2616	0.06752	1.2230	0.4489	0.011010	0.04272	0.02737	0.06041	0.1534	0.3391	0.2200	0
172	0	16	16.60	28.08	0.08455	0.1023	0.1590	0.05648	0.4564	1.0750	0.005903	0.03731	0.01557	0.01318	0.1139	0.3094	0.1418	0

138 rows × 19 columns

In [41]:
```python
# removed correlated  features
X_train.drop(labels=corr_features, axis=1, inplace=True)
X_test.drop(labels=corr_features, axis=1, inplace=True)

X_train.shape, X_test.shape
```

Out[41]: ((138, 19), (60, 19))

In [43]:
```python
main_list = ['classLabel', 'col3', 'col4', 'col5', 'col8', 'col9', 'col12', 'col13',
       'col14', 'col15','PredictLabel2']

df_final = best_df[main_list]
```

Modelling

In [44]:
```python
df_final
```

Out[44]:

	classLabel	col3	col4	col5	col8	col9	col12	col13	col14	col15	PredictLabel2
0	0	31	18.02	27.60	0.09489	0.10360	0.1865	0.06333	0.6249	1.8900	5
1	0	61	17.99	10.38	0.11840	0.27760	0.2419	0.07871	1.0950	0.9053	2
2	0	116	21.37	17.44	0.08836	0.11890	0.2333	0.06010	0.5854	0.6105	0
3	0	123	11.42	20.38	0.14250	0.28390	0.2597	0.09744	0.4956	1.1560	0
4	1	27	20.29	14.34	0.10030	0.13280	0.1809	0.05883	0.7572	0.7813	0
...	...	...	...	...	...	...	...	...	...	...	...
193	0	10	22.52	21.92	0.07592	0.09162	0.1728	0.05262	1.3740	2.3120	2
194	0	8	15.44	31.18	0.09399	0.10620	0.1735	0.06105	0.3235	1.8390	0
195	0	12	17.17	29.19	0.08952	0.06655	0.1793	0.05392	0.6101	1.4250	0
196	1	3	21.42	22.84	0.10700	0.19390	0.1884	0.06472	1.0850	0.8469	?
197	0	6	16.70	28.13	0.08896	0.11310	0.1890	0.06035	0.6052	1.2350	0

198 rows × 11 columns

In [67]:
```
# blood_glucose_random blood_urea serum_creatinine sodium potassium haemoglobin packed_cell_volume

X = df_final[["col12"]]
Y = df_final[["col8"]]
```

In [68]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y) #, test_size=0.3, random_state=101)

from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X_train, y_train)
```

Out[68]: LinearRegression()

In [69]:
```
#Training Accuracy
linreg.score(X_train,y_train)

#Prediction
prediction=linreg.predict(X_test)
```

In [70]:
```
from sklearn import metrics
print("MAE: ",metrics.mean_absolute_error(y_test,prediction))
print("MSE: ",metrics.mean_squared_error(y_test,prediction))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
MAE:  0.008347858189671122
MSE:  0.00010924328230945O1
RMSE:  0.010451951124524555
```

In [71]:
```
coef=pd.DataFrame()
coef['Features'] = X.columns.values
coef['Coefficients'] = linreg.coef_
coef
```

Out[71]:

	Features	Coefficients
0	col12	0.24215

In [72]:
```
#Testing Accuracy
linreg.score(X_test,y_test)
```

Out[72]: 0.32435067673638807

In [ ]: