AI BASED

DIABETES PREDICTION SYSTEM

PHASE-4

FEATURE ENGINEERING

- **Processing for Missing Values and Outliers**
- **Creating New Feature Interactions**

Processing for Missing Values and Outliers

```
na_cols = missing_values_table(df, True)
```

	n_miss	ratio
Insulin	374	48.70
SkinThickness	227	29.56
BloodPressure	35	4.56
BMI	11	1.43
Glucose	5	0.65

1. Define a Function about comparing target variable with missing values
2. Fill the missing values of some variables with the median
3. Fill the missing values of *Insulin* and *Skin Thickness* variables
4. Standardization of variables
5. Implement the KNN method
6. Undo the standardization of these variables

Define a Function about comparing target variable with missing values

```
def missing_vs_target(dataframe, target, na_columns):
 temp_df = dataframe.copy()
for col in  na_columns:
   temp_df[col + '_NA_FLAG'] = np.where(temp_df[col].isnull(), 1, 0)
 na_flags = temp_df.loc[:, temp_df.columns.str.contains("_NA_")].columns

 for col in  na_flags:
 print(pd.DataFrame({"TARGET_MEAN":
temp_df.groupby(col)[target].mean(),
 "Count": temp_df.groupby(col)[target].count()}), end="\n\n\n")

missing_vs_target(df, "Outcome", na_cols)
```

Glucose_NA_FLAG	TARGET_MEAN	Count
0	0.348624	763
1	0.400000	

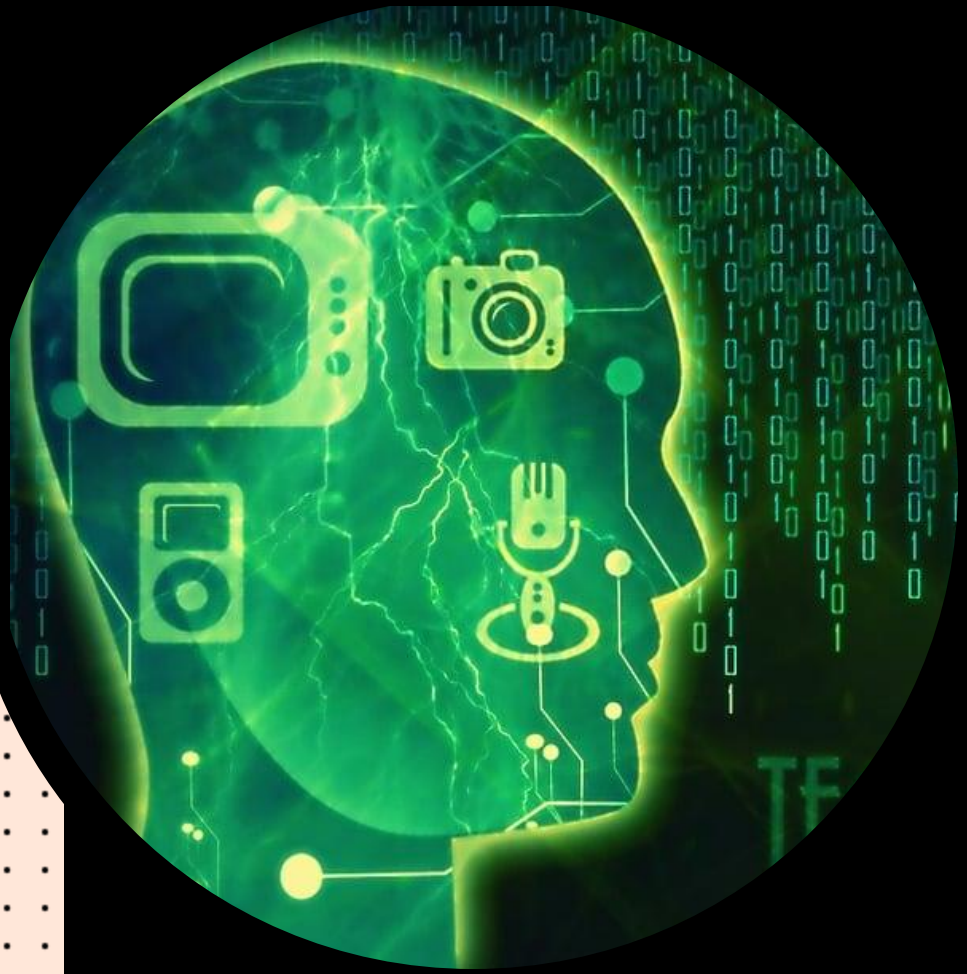
BloodPressure_NA_FLAG	TARGET_MEAN	Count
0	0.343793	733
1	0.457143	35

SkinThickness_NA_FLAG	TARGET_MEAN	Count
0	0.332717	541
1	0.387665	22

Fill the missing values of some variables with the median

```
df["glucose"] =
df["glucose"].fill
na(df["glucose"].media
n())
df["bloodpressure"] =
df["bloodpressure"].fill
na(df["bloodpressure"].
median()) df["bmi"] =
df["bmi"].fillna(df["bmi
"].median())
```

Standardization of variables



```
scaler = MinMaxScaler()
dff =
pd.DataFrame(scaler.fit_transform(dff),
columns=dff.columns)
dff.head
```

	Insulin	Skin Thickness
0	NaN	0.56
1	NaN	0.44
2	NaN	NaN
3	0.230797	0.32
4	0.444284	0.56

Implement The KNN Method

	Insulin	SkinThickness
0	0.591201	0.560000
1	0.477173	0.440000
2	0.382257	0.441183
3	0.230797	0.320000
4	0.444284	0.560000

```
from sklearn.impute
import KNNImputer
imputer = KNNImputer(n_neighbors=5):

dff =
pd.DataFrame(imputer.fit_transform(dff),
columns=dff.columns)
dff.head()
```

Undo the standardization of these variables

```
dff = pd.DataFrame(scaler.inverse_trans
form(dff), columns=dff.columns)
dff.head()
```

	Insulin	SkinThickness
0	218.925	35.00000
1	179.400	29.00000
2	146.500	29.05915
3	94.000	23.00000
4	168.000	35.00000

```
dff = pd.DataFrame(scaler.inverse_transform(dff),
columns=dff.columns)
dff.head()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

:

IN [40]:

Create a Insulin Categorical variable

1

Create a Glucose Categorical variable

2

Create the Age Categorical variable

3

Create the BMI Categorical variable

4

Create a Diastolic Blood Pressure Categorical variable

5

Create a Insulin Categorical variable

Create a Glucose Categorical variable

```
df.loc[(df['Glucose'] < 70), 'GLUCOSE_CAT'] ="hipoglisemi"
df.loc[(df['Glucose'] >= 70) & (df['Glucose'] < 100) , 'GLUCOSE_CAT']
="normal" df.loc[(df['Glucose'] >= 100) & (df['Glucose'] < 126) ,
'GLUCOSE_CAT'] ="imparied glucose" df.loc[(df['Glucose'] >= 126),
'GLUCOSE_CAT'] ="hiperglisemi"

df.groupby("GLUCOSE_CAT").agg({"Outcome": ["mean","count"]})
```

Outcome		
	mean	count
GLUCOSE_CAT		
hiperglisemi	0.592593	297
hipoglisemi	0.000000	11
imparied glucose	0.279570	279
normal	0.077348	181

Create the Age Categorical variable

```
df.loc[(df['Age'] >= 18) & (df['Age'] < 30) , 'AGE_CAT'] ="young_women_"
df.loc[(df['Age'] >= 30) & (df['Age'] < 45) , 'AGE_CAT'] ="mature_women"
df.loc[(df['Age'] >= 45) & (df['Age'] < 65) , 'AGE_CAT'] ="middle_age"
df.loc[(df['Age'] >= 65) & (df['Age'] < 75) , 'AGE_CAT'] ="old_age"
df.loc[(df['Age'] >= 75) , 'AGE_CAT'] ="elder_age"

df.groupby("AGE_CAT").agg({"Outcome": ["mean","count"]})
```

Outcome

	mean	count
AGE_CAT		
mature_women	0.493724	239
middle_age	0.529915	117
old_age	0.250000	16
young_women_	0.212121	396

CREATE THE BMI CATEGORICAL VARIABLE

BMI_CAT		
1st_Obese	0.438298	235
2nd_Obese	0.452830	212
3rd_Obese	0.611111	36
normal	0.068627	102
overweight	0.223464	179
weak	0.000000	4

```
df.loc[(df['BMI'] < 16), 'BMI_CAT'] ="overweak"
df.loc[(df['BMI'] >= 16) & (df['BMI'] < 18.5) , 'BMI_CAT'] ="weak"
df.loc[(df['BMI'] >= 18.5) & (df['BMI'] < 25) , 'BMI_CAT'] ="normal"
df.loc[(df['BMI'] >= 25) & (df['BMI'] < 30) , 'BMI_CAT'] ="overweight"
df.loc[(df['BMI'] >= 30) & (df['BMI'] < 35) , 'BMI_CAT'] ="1st_Obese"
df.loc[(df['BMI'] >= 35) & (df['BMI'] < 45) , 'BMI_CAT']
="2nd_Obese"
df.loc[(df['BMI'] >= 45), 'BMI_CAT'] ="3rd_Obese"
```

:

```
df.groupby("BMI_CAT").agg({"Outcome": ["mean","count"]})
```

CREATE A DIASTOLIC BLOOD PRESSURE CATEGORICAL VARIABLEI

```
df.loc[(df['BloodPressure']<70) ,'DIASTOLIC_CAT']="low"
df.loc[(df['BloodPressure']>=70)&(df['BMI']<90),
'DIASTOLIC_CAT']="normal"
df.loc[(df['BloodPressure']>=90),'DIASTOLIC_CAT']="high"
```

In [39]:

linkcode

```
df.groupby("DIASTOLIC_CAT").agg({"Outcome":
["mean","count"]})
```

Outcome

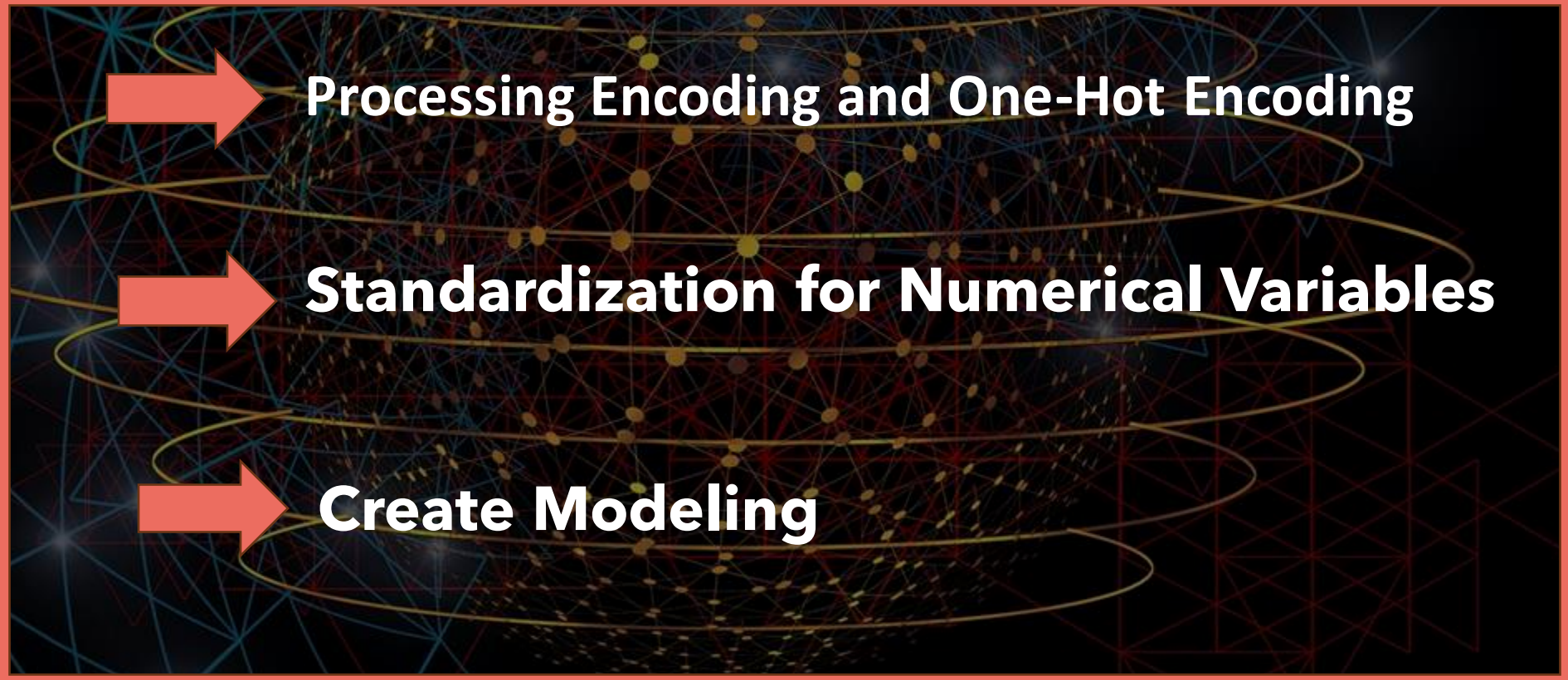
	mean	count
DIASTOLIC_CAT		
high	0.483333	60
low	0.247350	283
normal	0.397647	425

CREATE A INSULIN CATEGORICAL VARIABLE

Outcome		
	mean	count
INSULIN_CAT		
abnormal	0.429112	529
normal	0.171548	239

```
df.loc[(df['Insulin']< 120) ,
'INSULIN_CAT']="normal"
df.loc[(df['Insulin']>= 120),
'INSULIN_CAT']="abnormal"

In [41]:
df.groupby("INSULIN_CAT").agg({"Ou
tcome": ["mean","count"]})
```

MODELING


Processing Encoding and One-Hot Encoding
Standardization for Numerical Variables
Create Modeling

Processing Encoding and One-Hot Encoding

Label Encoder

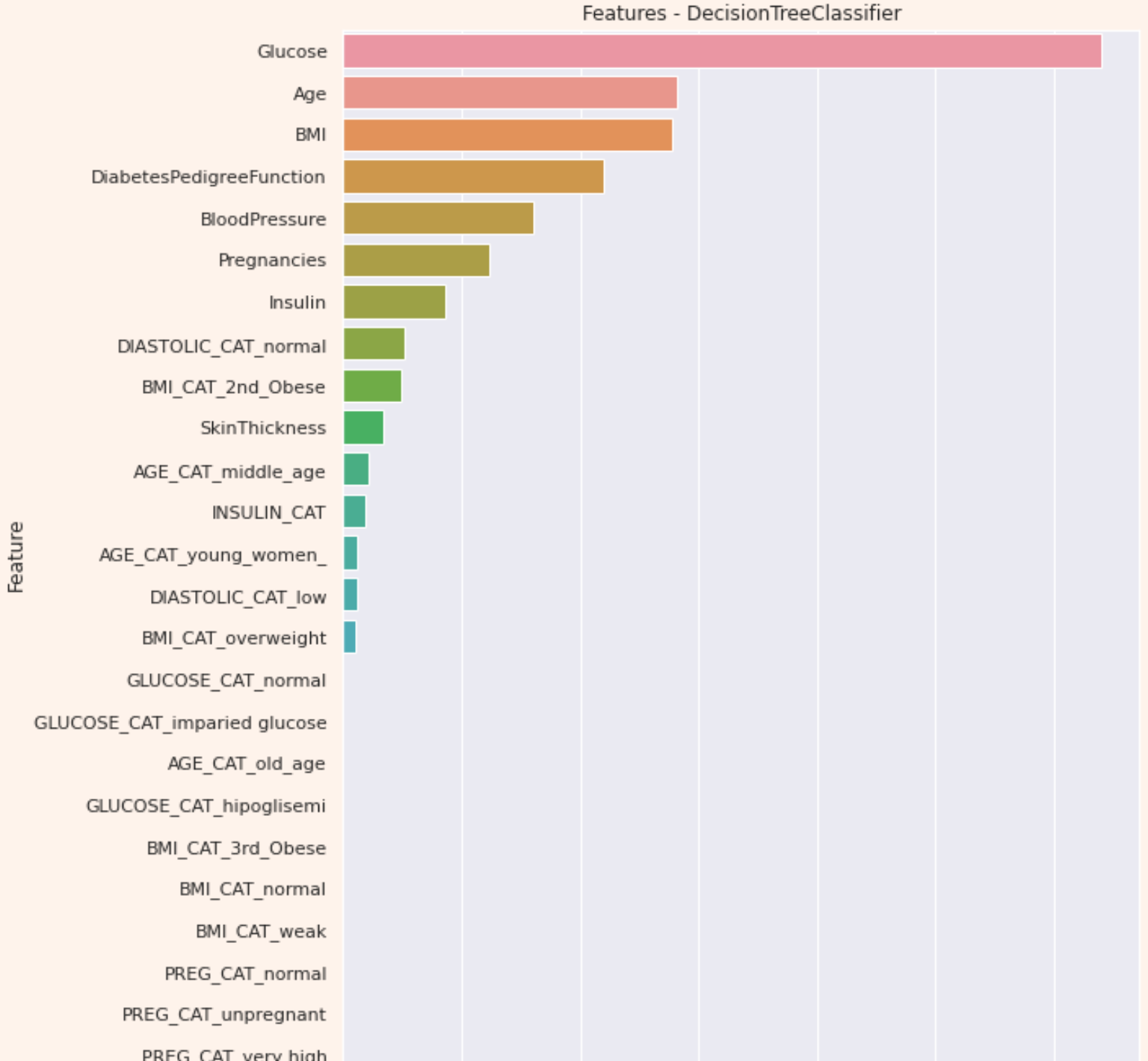
```
def label_encoder(dataframe, binary_col):
    labelencoder = LabelEncoder()
    dataframe[binary_col] = labelencoder.fit_transform(dataframe[binary_col])
    return dataframe

for col in binary_cols:
    df = label_encoder(df, col)
```

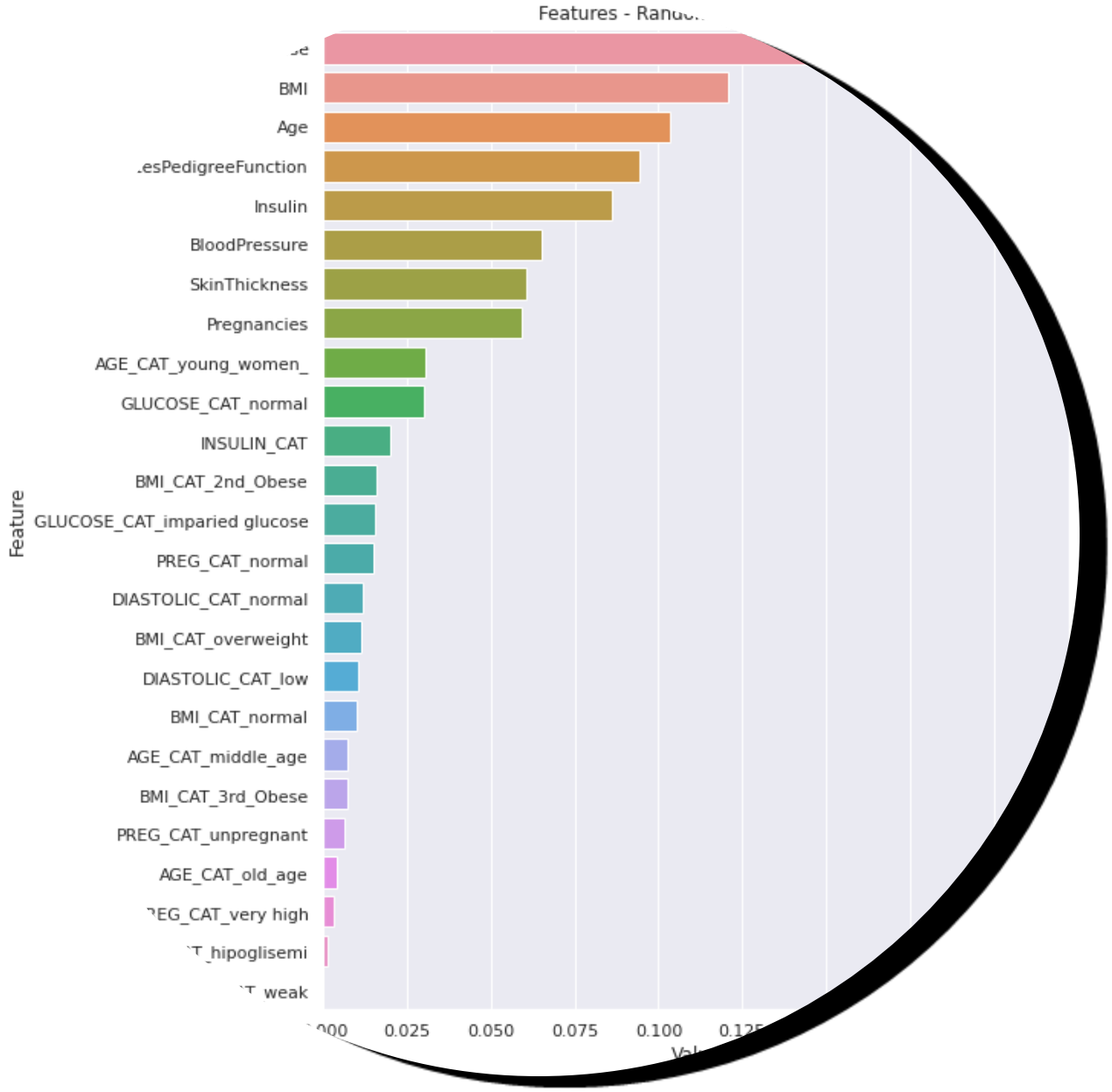
Define a Function about Label encoder

```
def one_hot_encoder(dataframe, categorical_cols, drop_first=True):
dataframe = pd.get_dummies(dataframe, columns=categorical_cols,
drop_first=drop_first) return dataframe ohe_cols = [col for col in
df.columns if 10 >= df[col].nunique() > 2]

df = one_hot_encoder(df, ohe_cols) df.head()
```

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	INSULIN_CAT	...	BMI_CAT_2nd_Obese	BMI_CAT_3rd_Obese	BMI_CAT_normal	BMI_CAT_overweight	BMI_CAT_weak	DIASTOLIC_CAT_low	DIASTOLIC_CAT_normal
0	0.647150	0.866045	-0.030632	7.044545e-01	1.064869	0.181092	0.588927	1.445691	1	0	...	0	0	0	0	0	0
1	-0.848970	-1.205066	-0.543914	-7.013859e-03	0.480186	-0.869465	-0.378101	-0.189304	0	0	...	0	0	0	1	0	1
2	1.245592	2.016666	-0.715005	-4.212732	-0.006496	-1.364725	0.746595	-0.103254	1	0	...	0	0	1	0	0	1

```
primitive_success=[]
model_names=[]
y=df['Outcome']
X=df.drop('Outcome',axis=1)
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30)

def ML(algName):
 # Model Building / Training
    model=algName().fit(X_train,y_train)
    model_name=algName.__name__
    model_names.append(model_name)
    # Prediction
    y_pred=model.predict(X_test)
```


Features - DecisionTreeClassifier
Feature
Glucose
Age
BMI
DiabetesPedigreeFunction
BloodPressure
Pregnancies
Insulin
DIASTOLIC_CAT_normal
BMI_CAT_2nd_Obese
SkinThickness
AGE_CAT_middle_age
INSULIN_CAT
AGE_CAT_young_women_
DIASTOLIC_CAT_low
BMI_CAT_overweight
GLUCOSE_CAT_normal
GLUCOSE_CAT_imparied glucose
AGE_CAT_old_age
GLUCOSE_CAT_hipoglisemi
BMI_CAT_3rd_Obese
BMI_CAT_normal
BMI_CAT_weak
PREG_CAT_normal
PREG_CAT_unpregnant
PREG_CAT_very high


Features - Rando
BMI
Age
esPedigreeFunction
Insulin
BloodPressure
SkinThickness
Pregnancies
AGE_CAT_young_women_
GLUCOSE_CAT_normal
INSULIN_CAT
BMI_CAT_2nd_Obese
GLUCOSE_CAT_imparied glucose
PREG_CAT_normal
DIASTOLIC_CAT_normal
BMI_CAT_overweight
DIASTOLIC_CAT_low
BMI_CAT_normal
AGE_CAT_middle_age
BMI_CAT_3rd_Obese
PREG_CAT_unpregnant
AGE_CAT_old_age
EG_CAT_very high
T_hipoglisemi
T_weak
Feature
000
0.025
0.050
0.075
0.100
0.125

- Imported Libraries And Read Diabetes Dataset
- Explorary Data Analysis : We Checked The Missing Values And We Defined A Function To Grab The Categorical And Numerical Variables Of Its Dataset. We Made The Target Variable Analysis And Outliers Analysis.
- Data Preprocessing : We Filled Missing Values Of Some Variables With Median Values Or The Knn Method.
- Featured Engineering: We Created New Feature Interactions For Categorical Variables.
- Encoding: One-hot-encoding Was Implemented For Categorical Variables.
- Modeling: We Created Ml Model For The Dataset. The Accuracy Score Was Calculated The Machine Learning Models That Are