In [63]:
```
import numpy as np
```

In [64]:
```
import pandas as pd
```

In [65]:
```
import matplotlib.pyplot as plt
```

In [66]:
```
import seaborn as sns
```

In [67]:
```
sns.set()
```

In [68]:
```
%matplotlib inline
```

In [69]:
```
diabetes_data = pd.read_csv('DESKTOP/diabetes.csv')
```

In [70]:
```
diabetes_data.head()
```

Out[70]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288

In [71]:
```
diabetes_data.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
```

```
Pregnancies                 768 non-null int64
Glucose                     768 non-null int64
BloodPressure               768 non-null int64
SkinThickness               768 non-null int64
Insulin                     768 non-null int64
BMI                         768 non-null float64
DiabetesPedigreeFunction    768 non-null float64
Age                         768 non-null int64
Outcome                     768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [72]:
```
diabetes_data.describe()
```

Out[72]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabetes
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [73]:
```
diabetes_data.describe().T
```

Out[73]:

	count	mean	std	min	25%	50%	75%
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000

	count	mean	std	min	25%	50%	75%
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000

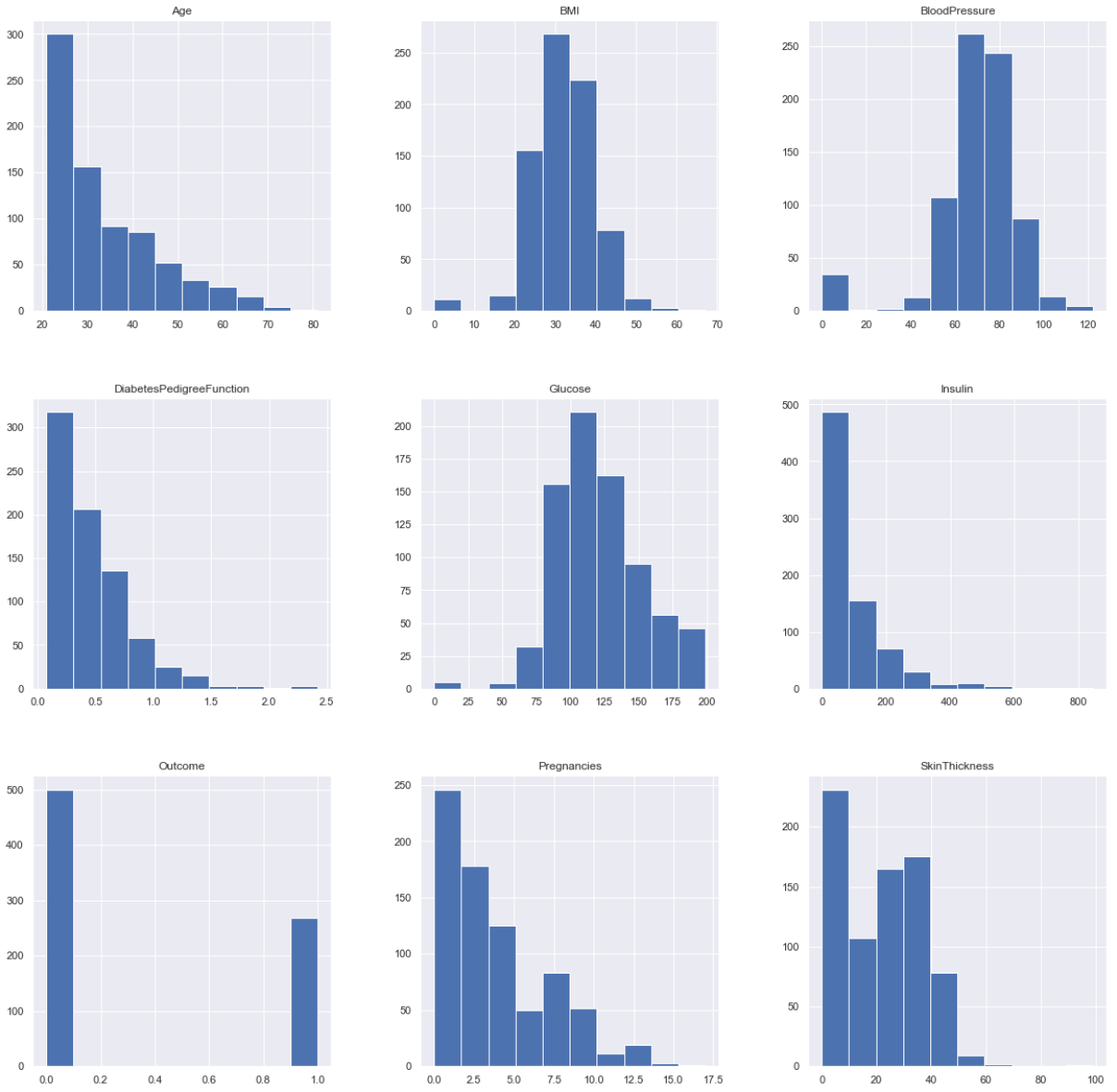
In [74]:
```
diabetes_data_copy = diabetes_data.copy(deep = True)
```

In [75]:
```
diabetes_data_copy[['Glucose','BloodPressure','SkinThickness','Insulin'
,'BMI']] = diabetes_data_copy[['Glucose','BloodPressure','SkinThicknes
s','Insulin','BMI']].replace(0,np.NaN)
```

In [76]:
```
print(diabetes_data_copy.isnull().sum())
```

```
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [77]:
```
p = diabetes_data.hist(figsize = (20,20))
```



In [78]:
```
diabetes_data_copy['Glucose'].fillna(diabetes_data_copy['Glucose'].mean
(), inplace = True)
```

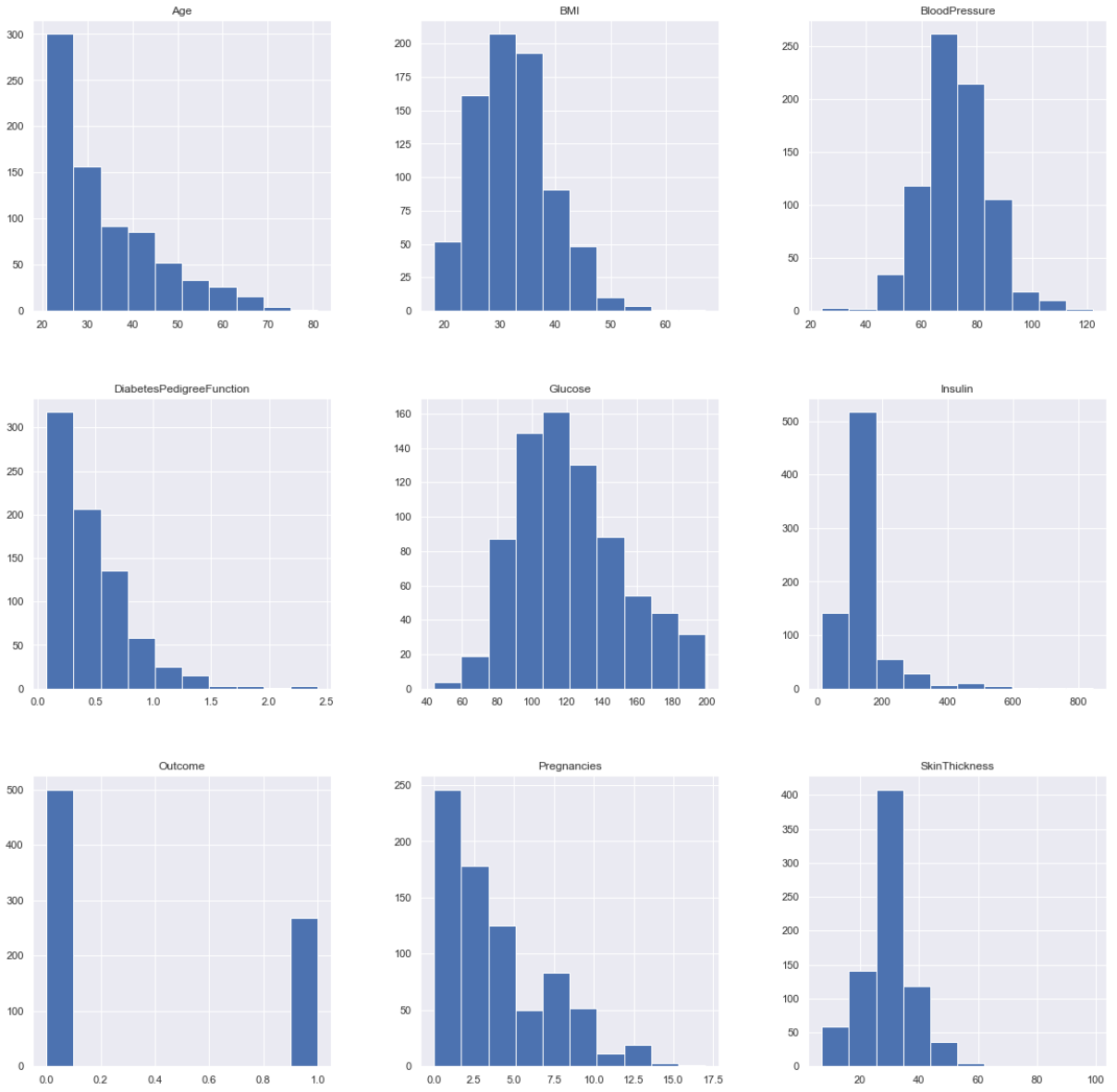
In [79]:
```
diabetes_data_copy['BloodPressure'].fillna(diabetes_data_copy['BloodPressure'].mean(), inplace = True)
```

In [80]:
```
diabetes_data_copy['SkinThickness'].fillna(diabetes_data_copy['SkinThickness'].median(), inplace = True)
```

In [81]:
```
diabetes_data_copy['Insulin'].fillna(diabetes_data_copy['Insulin'].median(), inplace = True)
```

In [82]:
```
diabetes_data_copy['BMI'].fillna(diabetes_data_copy['BMI'].median(), inplace = True)
```

In [83]:
```
p = diabetes_data_copy.hist(figsize = (20,20))
```



In [84]:
```
diabetes_data.shape
```

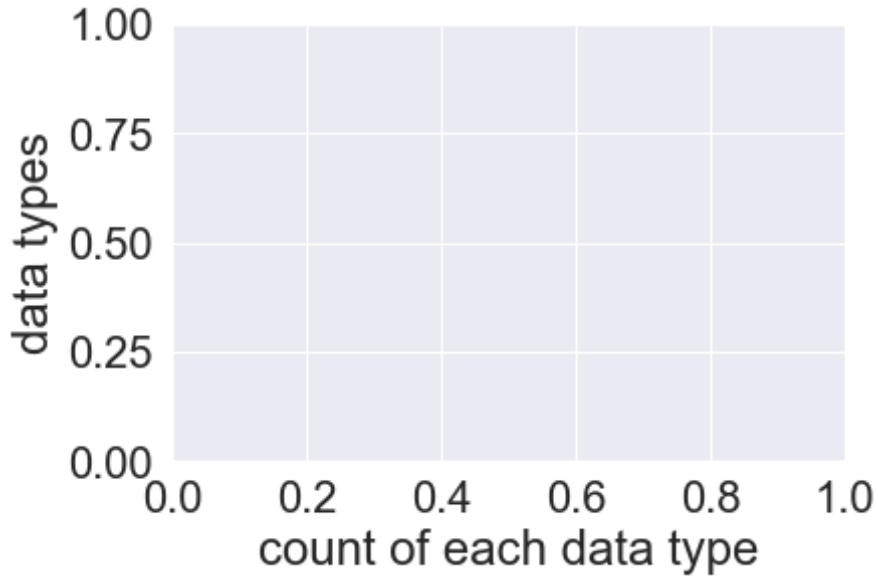
Out[84]: (768, 9)

In [85]:
```
plt.figure(figsize=(5,5))
sns.set(font_scale=2)
```

```
<Figure size 360x360 with 0 Axes>
```

In [86]:
```
plt.xlabel("count of each data type")
plt.ylabel("data types")
```

Out[86]:
```
Text(0, 0.5, 'data types')
```



In [87]:
```
plt.show()
```

In [88]:
```
from sklearn.preprocessing import StandardScaler
```

In [89]:
```
sc_X = StandardScaler()
```

In [90]:
```
X =  pd.DataFrame(sc_X.fit_transform(diabetes_data_copy.drop(["Outcome"
],axis = 1),),
        columns=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickn
```

```
ess', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age'])
X.head()
```

```
C:\Users\admin\Anaconda3\lib\site-packages\sklearn\preprocessing\data.p
y:625: DataConversionWarning: Data with input dtype int64, float64 were
all converted to float64 by StandardScaler.
  return self.partial_fit(X, y)
C:\Users\admin\Anaconda3\lib\site-packages\sklearn\base.py:462: DataCon
versionWarning: Data with input dtype int64, float64 were all converted
to float64 by StandardScaler.
  return self.fit(X, **fit_params).transform(X)
```

Out[90]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeI
0	0.639947	0.865108	-0.033518	0.670643	-0.181541	0.166619	
1	-0.844885	-1.206162	-0.529859	-0.012301	-0.181541	-0.852200	-(
2	1.233880	2.015813	-0.695306	-0.012301	-0.181541	-1.332500	
3	-0.844885	-1.074652	-0.529859	-0.695245	-0.540642	-0.633881	-(
4	-1.141852	0.503458	-2.680669	0.670643	0.316566	1.549303	

In [91]:
```
y = diabetes_data_copy.Outcome
```

In [92]:
```
from sklearn.model_selection import train_test_split
```

In [93]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=1/3,rand
om_state=42, stratify=y)
```

In [94]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [95]:
```
test_scores = []
```

In [96]:
```
train_scores = []
```

In [97]:
```
for i in range(1,15):

    knn = KNeighborsClassifier(i)
    knn.fit(X_train,y_train)

    train_scores.append(knn.score(X_train,y_train))
    test_scores.append(knn.score(X_test,y_test))
```

In [98]:
```
max_train_score = max(train_scores)
```

In [99]:
```
train_scores_ind = [i for i, v in enumerate(train_scores) if v == max_t
rain_score]
```

In [100]:
```
print('Max train score {} % and k = {}'.format(max_train_score*100,list
(map(lambda x: x+1, train_scores_ind))))
```

```
Max train score 100.0 % and k = [1]
```

In [101]:
```
max_test_score = max(test_scores)
```

In [102]:
```
test_scores_ind = [i for i, v in enumerate(test_scores) if v == max_tes
t_score]
```

In [103]:
```
print('Max test score {} % and k = {}'.format(max_test_score*100,list(m
ap(lambda x: x+1, test_scores_ind))))
```

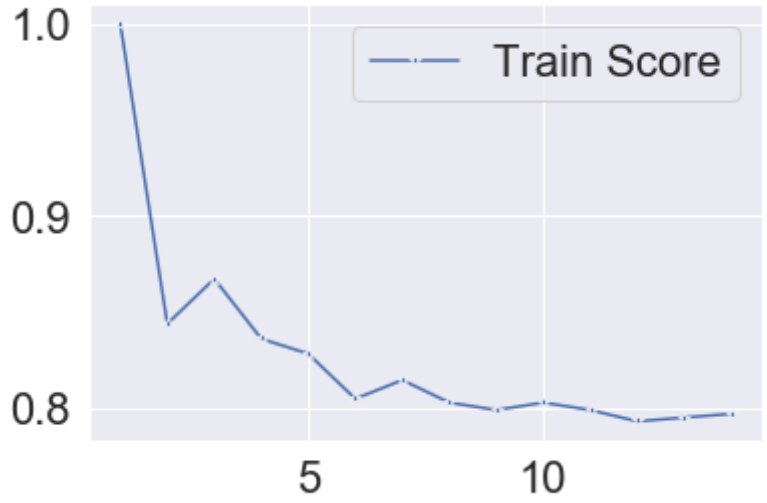
```
Max test score 76.5625 % and k = [11]
```

In [104]:
```
plt.figure(figsize=(12,5))
```

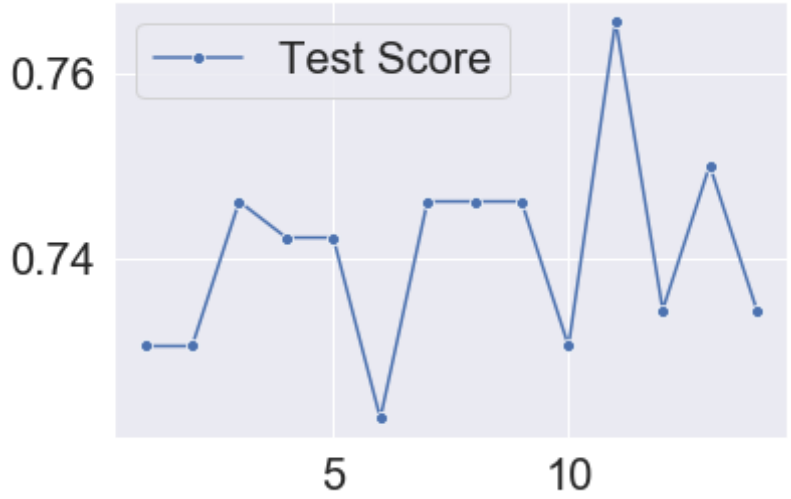
Out[104]:
```
<Figure size 864x360 with 0 Axes>

<Figure size 864x360 with 0 Axes>
```

In [105]:
```
p = sns.lineplot(range(1,15),train_scores,marker='*',label='Train Scor
e')
```



In [106]:
```
p = sns.lineplot(range(1,15),test_scores,marker='o',label='Test Score')
```



In [107]:
```
knn = KNeighborsClassifier(11)
```

In [108]:
```
knn.fit(X_train,y_train)
```

Out[108]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
i',
                     metric_params=None, n_jobs=None, n_neighbors=11, p=2,
                     weights='uniform')
```

In [109]:
```
knn.score(X_test,y_test)
```

Out[109]:
```
0.765625
```