Photo by Sharon McCutcheon on Unsplash

How to Predict the Onset of Diabetes Using Feature Selection and Correlation Matrix

To improve accuracy, avoid overfitting, train the model faster and reduce the complexity of the model

Abhinav Sagar

Oct 11, 2019 · 6 min read

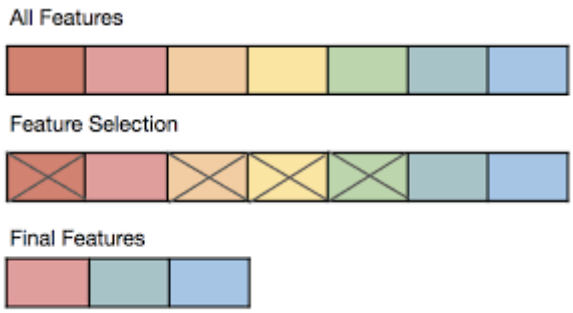
Stuck behind the paywall? Click here to read the full story with my Friend Link!

In a high dimensional dataset, there are some entirely irrelevant and unimportant features.The contribution of these types of features is negligible towards predictive modeling as compared to the important features. They may have zero contribution as well. These features cause a number of problems which in turn prevents the process of efficient predictive modeling. Some of the problems associated are:

1. Unnecessary time and memory consumption while training the model.
2. These features act as a noise for which the machine learning model can perform poorly. The model can learn the noise ie irrelevant features in the dataset and thus overfit.
3. It reduced accuracy of the predictive model.

So, what's the solution here? The best solution is Feature Selection.

Feature Selection is the process of selecting out the most significant features from a given dataset. In many of the cases, Feature Selection can enhance the performance of a machine learning model as well.



Feature Selection demonstration

The importance of feature selection:

1. It enables the machine learning algorithm to train faster.
2. It reduces the complexity of a model and makes it easier to interpret.
3. It improves the accuracy of a model if the right subset is chosen.
4. It reduces Overfitting.

. . .

In this article, I will demonstrate how feature selection can be used to reduce insignificant and unimportant features. I will be showing how feature importance and correlation matrix can be used for improving the machine learning model. I have used Pima Indians Diabetes Dataset for this project.

The Challenge

To diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset.

Environment and tools

1. scikit-learn
2. seaborn
3. numpy
4. pandas
5. matplotlib

Data

The dataset can be downloaded from the kaggle website which can be found here.

Description of variables in the dataset:

- **Pregnancies:** Number of times pregnant
- **Glucose:** Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- **BloodPressure:** Diastolic blood pressure (mm Hg)
- **SkinThickness:** Triceps skin fold thickness (mm)
- **Insulin:** 2-Hour serum insulin (mu U/ml)
- **BMI:** Body mass index (weight in kg/(height in m)^2)
- **DiabetesPedigreeFunction:** Diabetes pedigree function
- **Age:** Age (years)
- **Outcome:** Class variable (0 or 1)

Where is the code?

Without much ado, let's get started with the code. The complete project on github can be found here.

I started with loading all the libraries and dependencies required.

```
import pandas as pd
import numpy as np
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
import seaborn as sns
```

Let's see how the dataset looks like.

read_csv is a pandas function to read csv files and do operations on it later. **head()** method is used to return top n (5 by default) rows of a DataFrame.

```
df = pd.read_csv("../input/diabetes.csv")
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

shape() method is used to return the number of rows and columns present in the DataFrame.

```
df.shape
```

```
(599, 9)
```

columns() method is used to return all the column names present in the DataFrame.

```
df.columns
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

info() method is used to get a summary of the DataFrame. It is useful when doing exploratory data analysis (EDA) of the data.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 599 entries, 0 to 598
Data columns (total 9 columns):
Pregnancies                 599 non-null int64
Glucose                     599 non-null int64
BloodPressure               599 non-null int64
SkinThickness               599 non-null int64
Insulin                     599 non-null int64
BMI                         599 non-null float64
DiabetesPedigreeFunction    599 non-null float64
Age                         599 non-null int64
Outcome                     599 non-null int64
dtypes: float64(2), int64(7)
memory usage: 42.2 KB
```

describe() method computes a summary of statistics like count, mean, standard deviation, min, max and quartile values pertaining to the DataFrame columns.

```
df.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	599.000000	599.000000	599.000000	599.000000	599.000000	599.000000	599.000000	599.000000	599.000000
mean	3.824708	120.153589	68.732888	20.562604	79.460768	31.920033	0.481187	33.290484	0.347245
std	3.362839	32.682364	19.335675	16.017622	116.576176	8.008227	0.337552	11.828446	0.476492
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	64.000000	0.000000	0.000000	27.100000	0.248000	24.000000	0.000000
50%	3.000000	116.000000	70.000000	23.000000	36.000000	32.000000	0.383000	29.000000	0.000000
75%	6.000000	140.000000	80.000000	32.000000	123.500000	36.550000	0.647000	40.000000	1.000000
max	17.000000	198.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

sum() method of the DataFrame returned by **isnull()** method will give a series containing data about count of NaN in each column.

```
df.isnull().sum()
```

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

There are no null values in the dataset. Hence no data cleaning is required. Let's continue.

corr() method is used to find the pairwise correlation of all the columns in the DataFrame.

```
df.corr()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.139541	0.116680	-0.080902	-0.054812	0.033482	-0.052230	0.532545	0.207115
Glucose	0.139541	1.000000	0.143064	0.054430	0.339818	0.215949	0.144665	0.274057	0.449719
BloodPressure	0.116680	0.143064	1.000000	0.197535	0.095281	0.269482	0.008318	0.225625	0.061086
SkinThickness	-0.080902	0.054430	0.197535	1.000000	0.429068	0.377950	0.176592	-0.121553	0.075585
Insulin	-0.054812	0.339818	0.095281	0.429068	1.000000	0.184747	0.218313	-0.011523	0.145892
BMI	0.033482	0.215949	0.269482	0.377950	0.184747	1.000000	0.127675	0.046117	0.315894
DiabetesPedigreeFunction	-0.052230	0.144665	0.008318	0.176592	0.218313	0.127675	1.000000	0.033567	0.181561
Age	0.532545	0.274057	0.225625	-0.121553	-0.011523	0.046117	0.033567	1.000000	0.210234
Outcome	0.207115	0.449719	0.061086	0.075585	0.145892	0.315894	0.181561	0.210234	1.000000

Statistical tests can be used to select those features that have the strongest relationship with the output variable.

The below code is used to get the chi-squared statistical test to select all the features from the dataset. Also it assigns a score for every feature depending on the chi-squared test value.

```
X = df.iloc[:,0:8]
y = df.iloc[:,-1]
bestfeatures = SelectKBest(score_func=chi2, k=8)
```

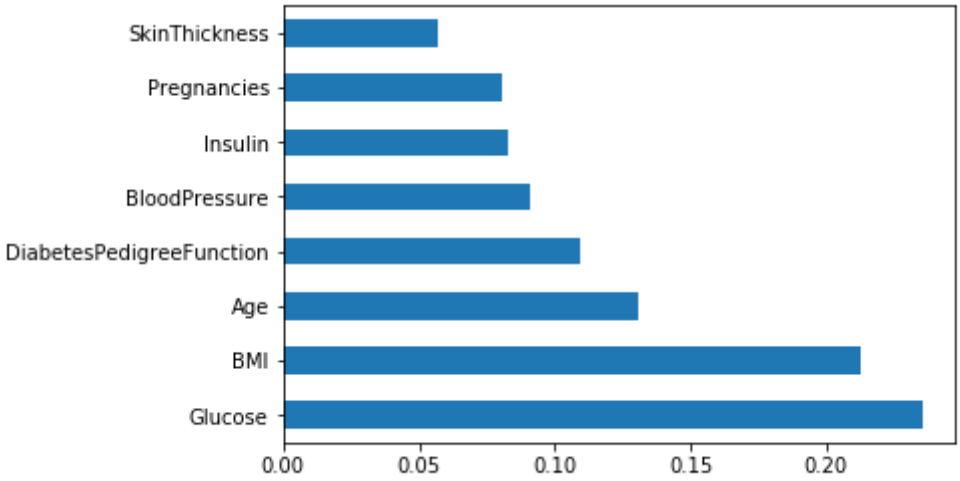
```
fit = bestfeatures.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Features','Score']
print(featureScores.nlargest(8,'Score'))
```

	Features	Score
4	Insulin	2176.855078
1	Glucose	1075.163239
5	BMI	119.892711
7	Age	111.081961
0	Pregnancies	75.846894
3	SkinThickness	42.627725
2	BloodPressure	12.137947
6	DiabetesPedigreeFunction	4.667840

Insulin has the highest score followed by glucose. This means that these features are more important for diabetes prediction than others.

Next, I compared feature importance using bar plots. Feature importance gives you a score for each feature present in the data. The higher the score, the more important or relevant is that feature towards the output variable. I have used Random Forest Classifier for ranking all the 8 features present in the dataset.

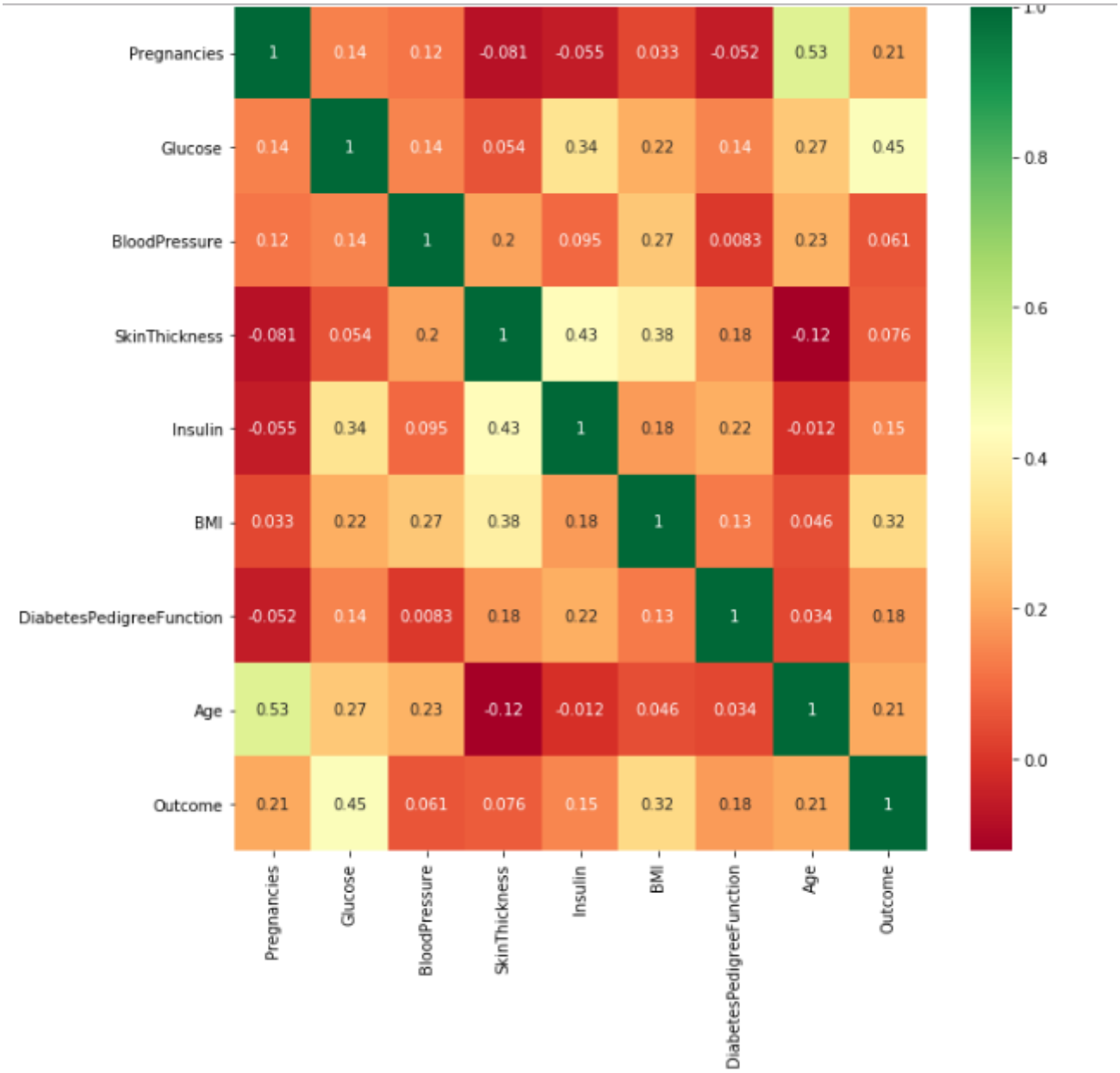
```
model = RandomForestClassifier()
model.fit(X,y)
print(model.feature_importances_)
feat_importances = pd.Series(model.feature_importances_,
index=X.columns)
feat_importances.nlargest(8).plot(kind='barh')
plt.show()
```



Glucose has the highest feature importance followed by body mass index. It is strange to see that insulin ranks much lower in this case.

Finally I made a correlation matrix to show how the features are related to each other or the target variable.Correlation can be positive (increase in one value of feature increases the value of the target variable) or negative (increase in one value of feature decreases the value of the target variable)

```
corrmat = df.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(10,10))
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



Conclusions

In this article, I started with the initial part of the data science workflow ie data exploration and data preparation. I continued with demonstrating how feature selection can be used to reduce insignificant and unimportant features by comparing the importance of features. Finally I concluded with making a correlation matrix for showing correlation coefficients among all the features.

References/Further Readings

Machine Learning Workflow on Diabetes Data : Part 01

"Machine learning in a medical setting can help enhance medical diagnosis dramatically."

towardsdatascience.com

Feature Importance and Feature Selection With XGBoost in Python

A benefit of using ensembles of decision tree methods like gradient boosting is that they can automatically provide...

machinelearningmastery.com

What Are Feature Selection Techniques In Machine Learning?

Image for representation purpose Feature selection is the method of reducing data dimension while doing predictive...

analyticsindiamag.com

Before You Go

The corresponding source code can be found here.

abhinavsagar/Machine-learning-tutorials

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or...

github.com

Contacts

If you want to keep updated with my latest articles and projects follow me on Medium. These are some of my contacts details:

- Personal Website
- Linkedin
- Medium Profile
- GitHub
- Kaggle

Happy reading, happy learning and happy coding!