Machine Learning Project Checklist

**Frame the Problem**

* Define the objectives or possible outcomes of the project
* Discuss how the solution will be used
* Identify any current solutions
  + Strengths of current solution
  + Shortcomings of current solution
* Define will success be measured
* Compare to similar projects
* List and verify assumptions

**Get the Data**

* Identify type and amount of data you will need
* Document where the data is located
* Confirm storage space needed
* Obtain any authorizations to access data
* Create workspace
* Get the data
* Ensure sensitive information is protected
* Inspect the data for size and type (time series, geographical, etc)
* Create a test set (train\_test\_split) early in the process

**Explore the Data**

* Create a copy of exploratory data analysis
  + Sample a smaller random set if needed
* Keep a record of the data exploration (Jupyter Notebook works great for this!)
* Study each feature
  + Type
  + Missing values
  + Usefulness
  + Distribution
* Identify target features
* Conduct data visualization
* Check for correlations between features
* Identify helpful data transformations
* Discuss other data that would useful
* Document lessons learned

**Wrangle the Data**

* Make a copy of the original dataset
* Consider using functions for data transformations
  + Easier to split tasks between team members
  + Reuse functions to clean / prep the test dataset
  + Can reuse logic on future datasets / projects
* Missing data & outliers
  + Fix, drop, replace, or accept
* Drop features that provide no useful information
* Feature engineering
  + Transform
  + Aggregate
  + Extract
* Standardize or normalize features

**Identify Promising Models**

* Train multiple models from different categories (linear, naïve Bayes, SVM, Random Forest, etc) using standard parameters
* Measure and compare model performance
* Identify significant variables for each model
* Analyze model errors
* Revise feature selection and feature engineering
  + Iterate model training & comparison as features change
* Identify the top models

**Tune the Models**

* Tune the hyperparameters
  + Treat data transformations as hyperparameters (replacing missing values with median, dropna, etc)
* Consider combining the best models
* Measure performance of final model

**Present Solution**

* Document process such that other data scientists could recreate your results
* Develop presentation
  + Highlight the big picture
  + Overview of data collected
  + Review analysis (at an appropriate level for audience)
  + Explain your solution
* Map solution to business objectives
* Interesting facts
  + Discuss what worked
  + Discuss what did not work
  + Assumptions
  + Surprises
  + Limitations
* Key findings are best represented visually
  + Charts
  + Graphs
  + Animations

**Launch**

* Prepare for deployment to a production environment
  + Modify code for production data inputs
  + Write unit tests
* Automate the monitoring of the production model
  + Reduction in performance
  + Input quality
* Periodically retrain model with fresh data

df\_ckd\_1 = pd.get\_dummies(df\_ckd\_1, columns = ['sg'])

df\_ckd\_1.rename(columns={'fuel-type\_gas':'gas', 'fuel-type\_diesel':'diesel'}, inplace=True)

df\_auto.head()

df\_data = df\_data[df\_data.rc != '\t?']df\_data.dropna(subset = ["wbc"], inplace=True)df\_data['rc'] = df\_data['rc'].astype('float')  
df\_data.reset\_index(drop=True, inplace=True)

#df\_data.info()#print(df\_data)s=df\_data['rc']

for i in range(269):  
if df\_data.iloc[i]['rc'] > 3.92 and df\_data.iloc[i]['rc'] < 5.65:  
df\_data.iloc[i, df\_data.columns.get\_loc('rbc')] = 'normal'  
  
else:  
df\_data.iloc[i, df\_data.columns.get\_loc('rbc')] = 'abnormal'df\_[data.to](http://data.to/)\_csv('output.csv')

**import** **scipy.stats** **as** **stats**

:

dataset\_table=pd.crosstab(dataset['sex'],dataset['smoker'])

print(dataset\_table)

#RBC

df\_table\_rbc = pd.crosstab(df\_ckd\_1['CKD'],df\_ckd\_1['rbc'])

print(df\_table\_rbc)

df\_table\_rbc.values

Observed\_values\_rbc = df\_table\_rbc.values

print("Observed Values :-\n",Observed\_values\_rbc)

val\_rbc=stats.chi2\_contingency(df\_table\_rbc)

val\_rbc

expected\_values\_rbc=val\_rbc[3]

no\_of\_rows\_rbc=len(df\_table\_rbc.iloc[0:2,0])

no\_of\_columns\_rbc=len(df\_table\_rbc.iloc[0,0:2])

ddof\_rbc=(no\_of\_rows-1)\*(no\_of\_columns-1)

print("Degree of Freedom:-",ddof)

alpha = 0.05

from scipy.stats import chi2

from scipy.stats import chi2

chi\_square=sum([(o-e)\*\*2./e for o,e in zip(Observed\_values,expected\_values)])

chi\_square\_statistic=chi\_square[0]+chi\_square[1]

print("chi-square statistic:-",chi\_square\_statistic)

#p-value

p\_value\_rbc=1-chi2.cdf(x=chi\_square\_statistic,df=ddof)

print('p-value:',p\_value\_rbc)

print('Significance level: ',alpha)

print('Degree of Freedom: ',ddof)

print('p-value:',p\_value\_rbc)

#PCC

df\_table\_pcc = pd.crosstab(df\_ckd\_1['CKD'],df\_ckd\_1['pcc'])

print(df\_table\_pcc)

Observed\_values\_pcc = df\_table\_pcc.values

val\_pcc=stats.chi2\_contingency(df\_table\_pcc)

expected\_values\_pcc=val[3]

no\_of\_rows\_pcc=len(df\_table\_pcc.iloc[0:2,0])

no\_of\_columns\_pcc=len(df\_table\_pcc.iloc[0,0:2])

ddof\_pcc=(no\_of\_rows-1)\*(no\_of\_columns-1)

print("Degree of Freedom:-",ddof)

alpha = 0.05

chi\_square\_pcc=sum([(o-e)\*\*2./e for o,e in zip(Observed\_values\_pcc,expected\_values\_pcc)])

chi\_square\_statistic\_pcc=chi\_square\_pcc[0]+chi\_square\_pcc[1]

print("chi-square statistic:-",chi\_square\_statistic\_pcc)

p\_value\_pcc=1-chi2.cdf(x=chi\_square\_statistic\_pcc,df=ddof)

print('p-value:',p\_value\_pcc)

print('Significance level: ',alpha)

print('Degree of Freedom: ',ddof)

print('p-value:',p\_value\_pcc)

#PC

df\_table\_pc = pd.crosstab(df\_ckd\_1['CKD'],df\_ckd\_1['pc'])

print(df\_table\_pc)

Observed\_values\_pc = df\_table\_pc.values

val\_pc=stats.chi2\_contingency(df\_table\_pc)

expected\_values\_pc=val[3]

no\_of\_rows\_pc=len(df\_table\_pc.iloc[0:2,0])

no\_of\_columns\_pc=len(df\_table\_pc.iloc[0,0:2])

ddof\_pc=(no\_of\_rows-1)\*(no\_of\_columns-1)

print("Degree of Freedom:-",ddof\_pc)

alpha = 0.05

chi\_square\_pc=sum([(o-e)\*\*2./e for o,e in zip(Observed\_values\_pc,expected\_values\_pc)])

chi\_square\_statistic\_pc=chi\_square\_pc[0]+chi\_square\_pc[1]

print("chi-square statistic:-",chi\_square\_statistic\_pc)

p\_value\_pc=1-chi2.cdf(x=chi\_square\_statistic\_pc,df=ddof)

print('p-value:',p\_value\_pc)

print('Significance level: ',alpha)

print('Degree of Freedom: ',ddof)

print('p-value:',p\_value\_pc)

#HTN

df\_table\_htn = pd.crosstab(df\_ckd\_1['CKD'],df\_ckd\_1['htn'])

print(df\_table\_htn)

Observed\_values\_htn = df\_table\_htn.values

val\_htn=stats.chi2\_contingency(df\_table\_htn)

expected\_values\_htn=val[3]

no\_of\_rows\_htn=len(df\_table\_htn.iloc[0:2,0])

no\_of\_columns\_htn=len(df\_table\_htn.iloc[0,0:2])

ddof\_htn=(no\_of\_rows-1)\*(no\_of\_columns-1)

print("Degree of Freedom:-",ddof\_htn)

alpha = 0.05

chi\_square\_htn=sum([(o-e)\*\*2./e for o,e in zip(Observed\_values\_htn,expected\_values\_htn)])

chi\_square\_statistic\_htn=chi\_square\_htn[0]+chi\_square\_htn[1]

print("chi-square statistic:-",chi\_square\_statistic\_htn)

p\_value\_htn=1-chi2.cdf(x=chi\_square\_statistic\_htn,df=ddof\_htn)

print('p-value:',p\_value\_htn)

print('Significance level: ',alpha)

print('Degree of Freedom: ',ddof\_htn)

print('p-value:',p\_value\_htn)

#DF

df\_table\_cad = pd.crosstab(df\_ckd\_1['CKD'],df\_ckd\_1['cad'])

print(df\_table\_cad)

Observed\_values\_cad = df\_table\_cad.values

val\_cad=stats.chi2\_contingency(df\_table\_cad)

expected\_values\_cad=val[3]

no\_of\_rows\_cad=len(df\_table\_cad.iloc[0:2,0])

no\_of\_columns\_cad=len(df\_table\_cad.iloc[0,0:2])

ddof\_cad=(no\_of\_rows-1)\*(no\_of\_columns-1)

print("Degree of Freedom:-",ddof\_cad)

alpha = 0.05

chi\_square\_cad=sum([(o-e)\*\*2./e for o,e in zip(Observed\_values\_cad,expected\_values\_cad)])

chi\_square\_statistic\_cad=chi\_square\_cad[0]+chi\_square\_cad[1]

print("chi-square statistic:-",chi\_square\_statistic\_cad)

p\_value\_cad=1-chi2.cdf(x=chi\_square\_statistic\_cad,df=ddof\_cad)

print('p-value:',p\_value\_cad)

print('Significance level: ',alpha)

print('Degree of Freedom: ',ddof\_cad)

print('p-value:',p\_value\_cad)

#DM

df\_ckd\_1["dm"].replace("\tno", "no", inplace=True)

df\_ckd\_1["dm"].replace("\tyes", "yes", inplace=True)

df\_table\_dm = pd.crosstab(df\_ckd\_1['CKD'],df\_ckd\_1['dm'])

print(df\_table\_dm)

Observed\_values\_dm = df\_table\_dm.values

val\_dm=stats.chi2\_contingency(df\_table\_dm)

expected\_values\_dm=val[3]

no\_of\_rows\_dm=len(df\_table\_dm.iloc[0:2,0])

no\_of\_columns\_dm=len(df\_table\_dm.iloc[0,0:2])

ddof\_dm=(no\_of\_rows-1)\*(no\_of\_columns-1)

print("Degree of Freedom:-",ddof\_dm)

alpha = 0.05

chi\_square\_dm=sum([(o-e)\*\*2./e for o,e in zip(Observed\_values\_dm,expected\_values\_dm)])

chi\_square\_statistic\_dm=chi\_square\_dm[0]+chi\_square\_dm[1]

print("chi-square statistic:-",chi\_square\_statistic\_dm)

p\_value\_dm=1-chi2.cdf(x=chi\_square\_statistic\_dm,df=ddof\_dm)

print('p-value:',p\_value\_dm)

print('Significance level: ',alpha)

print('Degree of Freedom: ',ddof\_dm)

print('p-value:',p\_value\_dm)

#APPET

df\_table\_appet = pd.crosstab(df\_ckd\_1['CKD'],df\_ckd\_1['appet'])

print(df\_table\_appet)

Observed\_values\_appet = df\_table\_appet.values

val\_appet=stats.chi2\_contingency(df\_table\_appet)

expected\_values\_appet=val[3]

no\_of\_rows\_appet=len(df\_table\_appet.iloc[0:2,0])

no\_of\_columns\_appet=len(df\_table\_appet.iloc[0,0:2])

ddof\_appet=(no\_of\_rows-1)\*(no\_of\_columns-1)

print("Degree of Freedom:-",ddof\_appet)

alpha = 0.05

chi\_square\_appet=sum([(o-e)\*\*2./e for o,e in zip(Observed\_values\_appet,expected\_values\_appet)])

chi\_square\_statistic\_appet=chi\_square\_appet[0]+chi\_square\_appet[1]

print("chi-square statistic:-",chi\_square\_statistic\_appet)

p\_value\_appet=1-chi2.cdf(x=chi\_square\_statistic\_appet,df=ddof\_appet)

print('p-value:',p\_value\_appet)

print('Significance level: ',alpha)

print('Degree of Freedom: ',ddof\_appet)

print('p-value:',p\_value\_appet)

#PE

df\_table\_pe = pd.crosstab(df\_ckd\_1['CKD'],df\_ckd\_1['pe'])

print(df\_table\_pe)

Observed\_values\_pe = df\_table\_pe.values

val\_pe=stats.chi2\_contingency(df\_table\_pe)

expected\_values\_pe=val[3]

no\_of\_rows\_pe=len(df\_table\_pe.iloc[0:2,0])

no\_of\_columns\_pe=len(df\_table\_pe.iloc[0,0:2])

ddof\_pe=(no\_of\_rows-1)\*(no\_of\_columns-1)

print("Degree of Freedom:-",ddof\_pe)

alpha = 0.05

chi\_square\_pe=sum([(o-e)\*\*2./e for o,e in zip(Observed\_values\_pe,expected\_values\_pe)])

chi\_square\_statistic\_pe=chi\_square\_pe[0]+chi\_square\_pe[1]

print("chi-square statistic:-",chi\_square\_statistic\_pe)

p\_value\_pe=1-chi2.cdf(x=chi\_square\_statistic\_pe,df=ddof\_pe)

print('p-value:',p\_value\_pe)

print('Significance level: ',alpha)

print('Degree of Freedom: ',ddof\_pe)

print('p-value:',p\_value\_pe)

#ANE

df\_table\_ane = pd.crosstab(df\_ckd\_1['CKD'],df\_ckd\_1['ane'])

print(df\_table\_ane)

Observed\_values\_ane = df\_table\_ane.values

val\_ane=stats.chi2\_contingency(df\_table\_ane)

expected\_values\_ane=val[3]

no\_of\_rows\_ane=len(df\_table\_ane.iloc[0:2,0])

no\_of\_columns\_ane=len(df\_table\_ane.iloc[0,0:2])

ddof\_ane=(no\_of\_rows-1)\*(no\_of\_columns-1)

print("Degree of Freedom:-",ddof\_ane)

alpha = 0.05

chi\_square\_ane=sum([(o-e)\*\*2./e for o,e in zip(Observed\_values\_ane,expected\_values\_ane)])

chi\_square\_statistic\_ane=chi\_square\_ane[0]+chi\_square\_ane[1]

print("chi-square statistic:-",chi\_square\_statistic\_ane)

p\_value\_ane=1-chi2.cdf(x=chi\_square\_statistic\_ane,df=ddof\_ane)

print('p-value:',p\_value\_ane)

print('Significance level: ',alpha)

print('Degree of Freedom: ',ddof\_ane)

print('p-value:',p\_value\_ane)

#BA

df\_table\_ba = pd.crosstab(df\_ckd\_1['CKD'],df\_ckd\_1['ba'])

print(df\_table\_ba)

Observed\_values\_ba = df\_table\_ba.values

val\_pe=stats.chi2\_contingency(df\_table\_ba)

expected\_values\_ba=val[3]

no\_of\_rows\_ba=len(df\_table\_ba.iloc[0:2,0])

no\_of\_columns\_ba=len(df\_table\_ba.iloc[0,0:2])

ddof\_ba=(no\_of\_rows-1)\*(no\_of\_columns-1)

print("Degree of Freedom:-",ddof\_ba)

alpha = 0.05

chi\_square\_ba=sum([(o-e)\*\*2./e for o,e in zip(Observed\_values\_ba,expected\_values\_ba)])

chi\_square\_statistic\_ba=chi\_square\_pe[0]+chi\_square\_ba[1]

print("chi-square statistic:-",chi\_square\_statistic\_ba)

p\_value\_ba=1-chi2.cdf(x=chi\_square\_statistic\_ba,df=ddof\_ba)

print('p-value:',p\_value\_ba)

print('Significance level: ',alpha)

print('Degree of Freedom: ',ddof\_ba)

print('p-value:',p\_value\_ba)

**from** **sklearn.naive\_bayes** **import** GaussianNB