Movie Recommender system:

* from nltk.stem.porter import PorterStemmer
* ps.stem('loved') -> 'love'
* from sklearn.feature\_extraction.text import CountVectorizer
* cv = CountVectorizer(max\_features=5000, stop\_words='english')
* from sklearn.metrics.pairwise import cosine\_similarity
* similarity = cosine\_similarity(vectors)

book-recommender-system:

* pivot\_table
* from sklearn.metrics.pairwise import cosine\_similarity
* similarity\_scores = cosine\_similarity(pt)

spam mail detection:

* from sklearn.preprocessing import LabelEncoder
* encoder = LabelEncoder()
* spam['target'] = encoder.fit\_transform(spam['target'])
* EDA (expotri data analysis)
* Nltk -> word\_tokenize, sent\_tokenize
* **Data processing**
* . Lower case
* . Tokenization
* . Removing special characters
* . Removing stop words and punctuation
* . Stemming
* from collections import Counter
* Counter(word\_len).most\_common(30)
* from sklearn.feature\_extraction.text import TfidfVectorizer
* tfidf = TfidfVectorizer()
* x = tfidf.fit\_transform(spam.tran\_text).toarray()
* from sklearn.naive\_bayes import MultinomialNB
* mnb = MultinomialNB()

car price predicter:

* from sklearn.linear\_model import LinearRegression
* from sklearn.preprocessing import OneHotEncoder
* from sklearn.compose import make\_column\_transformer
* from sklearn.pipeline import make\_pipeline
* from sklearn.metrics import r2\_score
* ohe=OneHotEncoder()
* ohe.fit(X[['name','company','fuel\_type']])
* column\_trans=make\_column\_transformer((OneHotEncoder(categories=ohe.categories\_),['name','company','fuel\_type']), remainder='passthrough')
* lr=LinearRegression()
* pipe=make\_pipeline(column\_trans,lr)
* pipe.steps[0][1].transformers[0][1].categories[0]

House price predictor:

* remove outliers by mean and std
* from sklearn.model\_selection import train\_test\_split
* from sklearn.linear\_model import LinearRegression, Lasso, Ridge
* from sklearn.preprocessing import OneHotEncoder, StandardScaler
* from sklearn.compose import make\_column\_transformer
* from sklearn.pipeline import make\_pipeline
* from sklearn.metrics import r2\_score
* column\_trans = make\_column\_transformer(
* (OneHotEncoder(sparse\_output=False), ['location']),
* remainder = 'passthrough')
* scaler = StandardScaler()
* lr = LinearRegression()
* pipe = make\_pipeline(column\_trans, scaler, ridge)

Face Mask Detection:

* from keras.preprocessing import image
* label coding logic
* from sklearn.model\_selection import train\_test\_split
* from keras.applications.vgg16 import VGG16
* from keras import Sequential
* model.compile(optimizer = 'Adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])
* model.fit(x\_train, y\_train, epochs= 15, validation\_data=(x\_test, y\_test))
* def detect\_face\_mask(img):
* y\_pred = model.predict(img.reshape(1, 224, 224, 3)/225.0)
* value = round(y\_pred[0,0]\*10,4)
* return 1 if value >= 0.01 else 0, value
* d = {
* 'name':[],
* 'label':[],
* 'predict':[]
* }
* for cate in categories:
* path = os.path.join('test', cate)
* label = categories.index(cate)
* for file in os.listdir(path)[10:20]:
* img\_path = os.path.join(path, file)
* d['name'].append(file)
* d['label'].append(label)
* smaple1 = cv2.imread(img\_path)
* image = cv2.resize(smaple1,(224,224))
* d['predict'].append(detect\_face\_mask(image))
* df = pd.DataFrame(d)
* df

Emotion Detection:

* from keras.utils import to\_categorical
* from keras\_preprocessing.image import load\_img,img\_to\_array
* from keras.models import Sequential
* from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D
* from tqdm.notebook import tqdm
* from sklearn.preprocessing import LabelEncoder
* model.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'] )
* model.fit(x= x\_train,y = y\_train, batch\_size = 128, epochs = 20, validation\_data = (x\_test,y\_test))
* model\_json = model.to\_json()
* with open("emotiondetector.json",'w') as json\_file:
* json\_file.write(model\_json)
* model.save("emotiondetector.h5")
* from keras.models import model\_from\_json
* json\_file = open("emotiondetector.json", "r")
* model\_json = json\_file.read()
* json\_file.close()
* model = model\_from\_json(model\_json)
* model.load\_weights("emotiondetector.h5")
* image = 'train/surprise/images - 2020-11-06T202524.557\_face.png'
* print("original image is of surprise")
* img = ef(image)
* pred = model.predict(img)
* pred\_label = label[pred.argmax()]
* print("model prediction is ",pred\_label)
* plt.imshow(img.reshape(48,48),cmap='gray')

Cat & Dog classification:

* import tensorflow as tf
* from tensorflow import keras
* from keras import Sequential
* from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, BatchNormalization, Dropout
* # Normalize
* def process(image,label):
* image = tf.cast(image/255. ,tf.float32)
* return image, label
* train\_ds = train\_ds.map(process)
* validation\_ds = validation\_ds.map(process)
* model.compile(optimizer='adam',loss='binary\_crossentropy',metrics=['accuracy'])
* history = model.fit(train\_ds,epochs=10,validation\_data=validation\_ds)
* import cv2
* test\_img = cv2.imread('/content/dog.jpeg')
* plt.imshow(test\_img)
* plt.show()
* test\_img = cv2.resize(test\_img,(256,256))
* test\_input = test\_img.reshape((1,256,256,3))
* model.predict(test\_input)