LECTURE 14 - MACHINE LEARNING REVIEW

STEPS TO BUILDING A MACHINE LEARNING MODEL

1. Load the Data
2. Clean the Data
3. Choose Target Column
4. Choose Predictor Columns
5. Decide Regression or Classification
6. Select Machine Learning Model
7. Build Machine Learning Model

1. LOAD THE DATA

```
import pandas as pd

file = 'good_link_or_CSV_file_uploaded_to_colab'

df = pd.read_csv(file)

df.head()
```

2. CLEAN THE DATA

A. Remove all null values.

B. Make sure data is all numeric.

A. CLEAN NULL VALUES

```
# show column info

df.info()

# replace individual column by value

df['null_col'] =
df['null_col].fillna(df['null_col].median())

# replace all null values by median

df = df.fillna(df.median())

# delete all null rows

df.dropna()
```

B. MAKE COLUMNS ALL NUMERIC

```
df = pd.get_dummies(df)
```

3. CHOOSE TARGET COLUMN

```
y = df['target_column']
```

4. CHOOSE PREDICTOR COLUMNS

```
X = df.drop('target_column', axis=1)

X = X.drop(['any', 'additional', 'columns'], axis=1)
```

5. DECIDE REGRESSION OR CLASSIFICATION

```
y.value_counts()

# Choose classification if target column is binary or
few limited options

# Choose regression if target column represents
continuous range of values
```

6. SELECT MACHINE LEARNING MODEL

```
# Regression
from xgboost import XGBRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
```

6. SELECT MACHINE LEARNING MODEL

```python
# Classification
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
```

7. BUILD MACHINE LEARNING MODEL

A. Split data into training and test set

B. Initialize model

C. Fit model on training set

D. Score model

A. SPLIT DATA INTO TRAINING AND TEST SET

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y)
```

B. INITIALIZE MODEL

```
model = XGBClassifier()
```

C. FIT MODEL ON TRAINING SET

```
model.fit(X_train, y_train)

# "model" is now a machine learning model
```

D. SCORE MODEL

```
# classification

from sklearn.metrics import accuracy_score

y_pred = model.predict(X_test)

accuracy_score(y_pred, y_test)

# there are many scoring metrics

# imbalanced data requires another metric
```

D. SCORE MODEL

```
# regression

from sklearn.metrics import mean_squared_error

y_pred = model.predict(X_test)

mse = mean_squared_error(y_pred, y_test)

mse**0.5

# alternative - r2_score (between 0 and 1)

model.score(X_test, y_test)
```

CROSS_VAL_SCORE

```
# more reliable way of scoring model

from sklearn.model_selection import cross_val_score

# classification

scores = cross_val_score(XGBClassifier(), X, y, cv=5)

scores.mean()

# regression

scores = cross_val_score(XGBRegressor(), X, y,
scoring='neg_mean_squared_error')

rmse_scores = (-scores)**0.5

rmse_scores.mean()
```

IMPROVING MODEL

A. More data

B. Try different models

C. Engineer new columns of data

D. Tune parameters

TUNE PARAMETERS

A. Examine model parameters

B. Adjust model parameters and score new model

C. Use GridSearchCV or RandomizedSearchCV to expedite process

TUNE PARAMETERS MANUALLY

```
# Check parameters and try different values

cross_val_score(XGBClassifier(max_depth=2), X, y)

cross_val_score(XGBClassifier(max_depth=4), X, y)
```

TUNE PARAMETERS GRIDSEARCHCV

```
from sklearn.model_selection import GridSearchCV

params = {'max_depth':[2, 4, 6, 8, 10, 20, 30]}

grid = GridSearchCV(XGBClassifier(), params,
scoring='accuracy')

grid.fit(X, y)

grid.best_params_

grid.best_score_
```

FINALIZE MODEL

```
# initialize model with any adjusted parameters

model = XGBClassifier(max_depth=4)

# fit model on all the data

model.fit(X, y)
```

MAKING PREDICTIONS WITH MODEL

```
# place new_data in pandas DataFrame with same X columns

X_new = new_data

model.predict(X_new)
```

BONUS - MOST IMPORTANT COLUMNS

```
# Show influence of columns for tree-based models

model.feature_importances_

# Show importance of columns in order

feature_dict = dict(zip(X.columns.to_list(),
model.feature_importances_))

import operator

sorted(feature_dict.items(), key=operator.itemgetter(1),
reverse=True)
```

LET'S CODE!

Go to

colab.research.google.com

and open a new notebook.