DataPreprocessing

April 15, 2020

1 Introduction

Source Dataset: https://www.kaggle.com/dipam7/student-grade-prediction

Data Preprocessing Steps: * Data Visualization * Data Transformation * Data Normalization * Feature Selection

1.0.1 Imports

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

import tensorflow_docs as tfdocs
import tensorflow_docs.modeling
import tensorflow_docs.plots
```

1.0.2 Load Dataset

[ ]:
```
raw_df = pd.read_csv("./student-mat.csv")
raw_df.head()
```

1.1 Dataset Metadata

1.1.1 Features

Nominal Category

- school - student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)
- sex - student's sex (binary: 'F' - female or 'M' - male)
- address - student's home address type (binary: 'U' - urban or 'R' - rural)
- famsize - family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than 3)

- Pstatus - parent's cohabitation status (binary: 'T' - living together or 'A' - apart)
- Mjob - mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
- Fjob - father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
- reason - reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')
- guardian - student's guardian (nominal: 'mother', 'father' or 'other')

Ordinal Category ('yes' => 1 has higher value than 'no' => 0)

- schoolsup - extra educational support (binary: yes or no)
- famsup - family educational support (binary: yes or no)
- paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
- activities - extra-curricular activities (binary: yes or no)
- nursery - attended nursery school (binary: yes or no)
- higher - wants to take higher education (binary: yes or no)
- internet - Internet access at home (binary: yes or no)
- romantic - with a romantic relationship (binary: yes or no)

Prepared Ordinal Category

- Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)
- Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)

Numeric

- age - student's age (numeric: from 15 to 22)
- traveltime - home to school travel time (numeric: 1 - 1 hour)
- studytime - weekly study time (numeric: 1 - 10 hours)
- failures - number of past class failures (numeric: n if 1<=n<3, else 4)
- famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
- freetime - free time after school (numeric: from 1 - very low to 5 - very high)
- goout - going out with friends (numeric: from 1 - very low to 5 - very high)
- Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
- Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
- health - current health status (numeric: from 1 - very bad to 5 - very good)
- absences - number of school absences (numeric: from 0 to 93)
- G1 - first period grade (numeric: from 0 to 20)
- G2 - second period grade (numeric: from 0 to 20)
- G3 - final grade (numeric: from 0 to 20)

1.1.2 Label

- avgGrade - Derived from (G1 + G2 + G3) / 3

[ ]:
```
raw_df.dtypes
```

[ ]:
```
df = raw_df.copy()
df['avgGrade'] = df.apply(lambda row: (row.G1 + row.G2 + row.G3)/3,axis=1)
df = df.drop(labels=['G1','G2','G3'],axis=1)
df.head()
```

[ ]:
```
plt.figure(figsize=(5,5))
corr = df.corr()
ax = sns.heatmap(corr, annot=True,annot_kws={"fontsize" : 1})
plt.show()
```

[ ]:
```
le = LabelEncoder()
ordinal = df.copy()
binary_feature =␣
 ↪['schoolsup','famsup','paid','activities','nursery','higher','internet','romantic']␣
 ↪#Ordinal Binary
ordinal[binary_feature] = ordinal[binary_feature].apply(lambda col: le.
 ↪fit_transform(col))
```

[ ]:
```

```

[ ]:
```
plt.figure(figsize=(5,5))
corr = ordinal.corr()
ax = sns.heatmap(corr, annot=False,annot_kws={"fontsize" : 9}, fmt=".1f")
plt.show()
```

[ ]:
```
normal = ordinal.copy()
notObject = normal.columns[normal.dtypes!=object].tolist()
notObject.remove('avgGrade')
normal[notObject]=(normal[notObject]-normal[notObject].min())/
 ↪(normal[notObject].max()-normal[notObject].min())
```

[ ]:
```
normal.head()
```

[ ]:
```
corr_target = abs(corr["avgGrade"])
bad_feature = corr_target[corr_target < 0.1]
print(bad_feature)
# print(babad_featureeature)
```

[ ]:
```
# dropped = normal.drop(labels=['famsup','paid','activities','nursery'
#                                                      ␣
 ↪,'famrel','freetime','Dalc','Walc','health','absences'],axis=1)
dropped = normal.copy()
dropped.head()
```

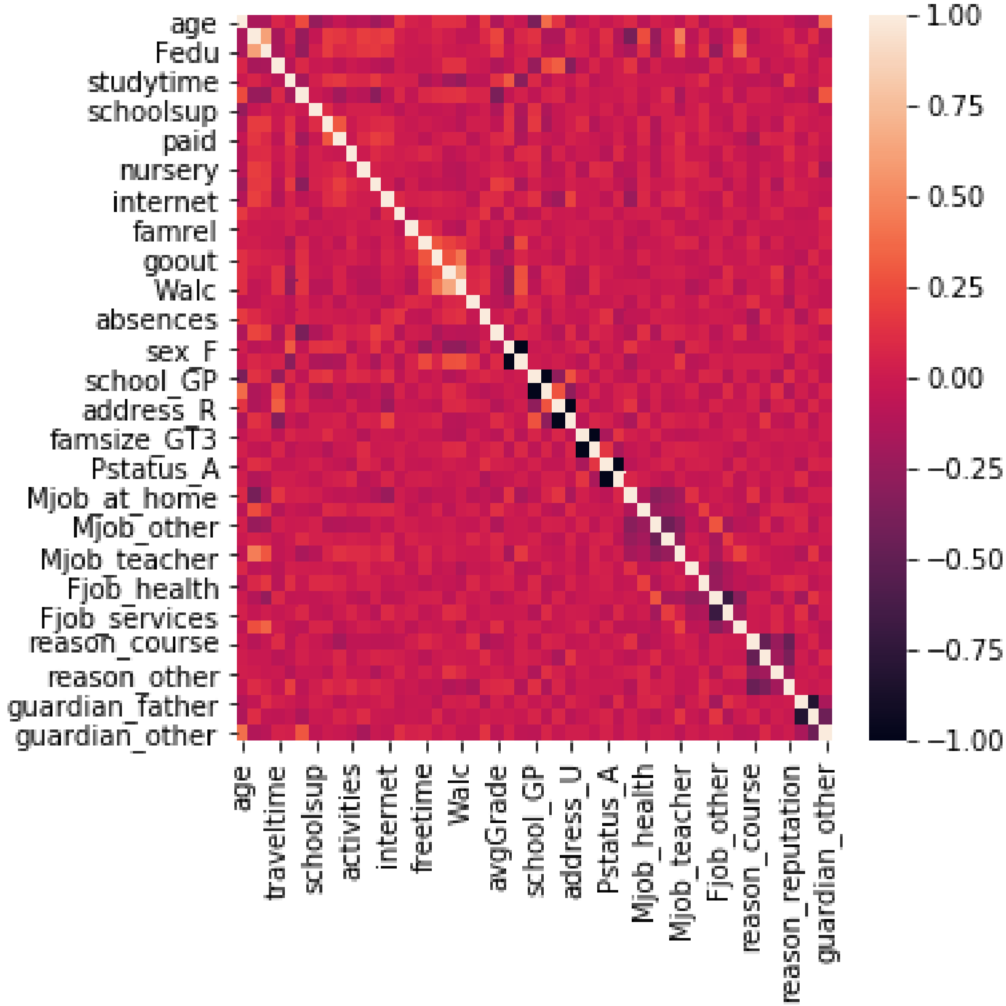
[ ]:
```
dropped.dtypes
```

[ ]:
```python
one_hot_categorical = dropped.columns[dropped.dtypes==object].tolist()
dropped[one_hot_categorical] = dropped[one_hot_categorical].astype('category')
print(one_hot_categorical)
```

[ ]:
```python
oh_concat = dropped
oh_concat = pd.concat([oh_concat,pd.
 ↪get_dummies(dropped['sex'],prefix='sex')],axis=1)
oh_concat = pd.concat([oh_concat,pd.
 ↪get_dummies(dropped['school'],prefix='school')],axis=1)
oh_concat = pd.concat([oh_concat,pd.
 ↪get_dummies(dropped['address'],prefix='address')],axis=1)
oh_concat = pd.concat([oh_concat,pd.
 ↪get_dummies(dropped['famsize'],prefix='famsize')],axis=1)
oh_concat = pd.concat([oh_concat,pd.
 ↪get_dummies(dropped['Pstatus'],prefix='Pstatus')],axis=1)
oh_concat = pd.concat([oh_concat,pd.
 ↪get_dummies(dropped['Mjob'],prefix='Mjob')],axis=1)
oh_concat = pd.concat([oh_concat,pd.
 ↪get_dummies(dropped['Fjob'],prefix='Fjob')],axis=1)
oh_concat = pd.concat([oh_concat,pd.
 ↪get_dummies(dropped['reason'],prefix='reason')],axis=1)
oh_concat = pd.concat([oh_concat,pd.
 ↪get_dummies(dropped['guardian'],prefix='guardian')],axis=1)
oh_concat = oh_concat.drop(labels=one_hot_categorical,axis=1)
```

[ ]:
```python
corr_target = abs(oh_concat.corr()["avgGrade"])
bad_feature = corr_target[corr_target < 0.1]
print(bad_feature)
# print(babad_featureeature)
```

[16]:
```python
plt.figure(figsize=(5,5))
corr = oh_concat.corr()
ax = sns.heatmap(corr, annot=False,annot_kws={"fontsize" : 2}, fmt=".1f")
plt.show()
oh_concat.to_csv('all_prepared.csv')
```



[17]:
```
oh_minimized = oh_concat.copy()
oh_minimized = oh_concat.
↪drop(labels=['school_GP','school_MS','Pstatus_A','Pstatus_T','Fjob_at_home','Fjob_health'
                                                ␣
↪,'Fjob_services','reason_home','reason_other','guardian_father','guardian_mother'],axis=1)
```

[18]:
```
oh_dropped = oh_concat.copy()
oh_dropped = oh_concat.
↪drop(labels=['school_GP','school_MS','famsize_GT3','famsize_LE3','Pstatus_A','Pstatus_T','M
                                        ␣
↪,'Mjob_teacher','Fjob_at_home','Fjob_health','Fjob_other','Fjob_services'
                                        ␣
↪,'reason_course','reason_home','reason_other','reason_reputation','guardian_father'
                             ,'guardian_mother','guardian_other'],axis=1)
```

```
[28]: #Raw Data + Label without categorical feature
phase1 = df.copy().drop(labels=df.columns[df.dtypes==object],axis=1) #mae: 3.
↪2583 - mse: 16.7854, early stop: mae: 2.9213 - mse: 12.6809

#Ordinal Category Added
phase2 = ordinal.copy().drop(labels=ordinal.columns[ordinal.
↪dtypes==object],axis=1) #mae: 3.4332 - mse: 17.3245, early: mae: 2.9872 -␣
↪mse: 13.4815

#Data Normalized
phase3 = normal.copy().drop(labels=normal.columns[normal.
↪dtypes==object],axis=1) #mae: 3.8822 - mse: 26.1232, early : mae: 3.0266 -␣
↪mse: 14.1503

#Feature Selection With CorrelationCoefficient > 0.1
phase4 = dropped.copy().drop(labels=dropped.columns[dropped.
↪dtypes=='category'],axis=1) #mae: 3.5513 - mse: 21.8263, early: mae: 3.2377␣
↪- mse: 15.4592

#One-Hot Encoding For Nominal Category
phase5 = oh_concat.copy() #mae: 3.7460 - mse: 23.5039, mae: 2.8213 - mse: 12.
↪1035

#phase5.5 --> Without Dropping in Phase 4, mae: 3.3277 - mse: 17.1304, es: mae:␣
↪2.9851 - mse: 13.8183

#Feature Selection With CorrelationCoefficient > 0.05
phase6 = oh_minimized.copy() #mae: 4.4667 - mse: 29.3205, early: mae: 2.8257 -␣
↪mse: 12.2229

#Feature Selection With CorrelationCoefficient > 0.1
phase7 = oh_dropped.copy() #mae: 3.9745 - mse: 23.1792, early: mae: 2.8674 -␣
↪mse: 12.6983

pil = phase5
# pil.dtypes
pil.head()
```

```
[28]:         age  Medu  Fedu  traveltime  studytime  failures  schoolsup  famsup  \
      0  0.428571  1.00  1.00    0.333333   0.333333       0.0        1.0     0.0
      1  0.285714  0.25  0.25    0.000000   0.333333       0.0        0.0     1.0
      2  0.000000  0.25  0.25    0.000000   0.333333       1.0        1.0     0.0
      3  0.000000  1.00  0.50    0.000000   0.666667       0.0        0.0     1.0
      4  0.142857  0.75  0.75    0.000000   0.333333       0.0        0.0     1.0

         paid  activities  …  Fjob_other  Fjob_services  Fjob_teacher  \
```

```
0   0.0          0.0  …          0             0             1
1   0.0          0.0  …          1             0             0
2   1.0          0.0  …          1             0             0
3   1.0          1.0  …          0             1             0
4   1.0          0.0  …          1             0             0

   reason_course  reason_home  reason_other  reason_reputation  \
0              1            0             0                  0
1              1            0             0                  0
2              0            0             1                  0
3              0            1             0                  0
4              0            1             0                  0

   guardian_father  guardian_mother  guardian_other
0                0                1               0
1                1                0               0
2                0                1               0
3                0                1               0
4                1                0               0

[5 rows x 49 columns]
```

[29]:
```
x_train = pil.copy().sample(frac=0.8,random_state=0)
x_test = pil.drop(x_train.index)
```

[30]:
```
y_train = x_train.pop('avgGrade')
y_test = x_test.pop('avgGrade')
```

[31]:
```
def build_model():

    model = keras.Sequential()
    model.add(layers.Dense(100, activation='relu',input_shape=[len(x_train.
 ↪columns)]))
    model.add(layers.Dense(100, activation='relu'))
    model.add(layers.Dense(1))

#     optimizer = tf.keras.optimizers.RMSprop(0.001)
    optimizer = keras.optimizers.Adam(learning_rate=0.001)

    model.compile(optimizer=optimizer, loss='mse', metrics=['mae','mse'])

    return model


a = keras.callbacks.EarlyStopping(
    monitor='val_loss', min_delta=0, patience=10, verbose=0, mode='auto'
)
```

```
model = build_model()
```

[32]:
```
model.fit(x_train, y_train, epochs=500, verbose=1, validation_split=0.2␣
↪,callbacks=[tfdocs.modeling.EpochDots(),a])
```

```
Train on 252 samples, validate on 64 samples
Epoch 1/500
 32/252 [==>…] - ETA: 3s - loss: 131.7852 - mae:
10.8927 - mse: 131.7852
Epoch: 0, loss:102.5905,  mae:9.3907,  mse:102.5905,  val_loss:99.1755,
val_mae:9.4202,  val_mse:99.1755,
252/252 [==============================] - 1s 2ms/sample - loss: 102.5905 - mae:
9.3907 - mse: 102.5905 - val_loss: 99.1755 - val_mae: 9.4202 - val_mse: 99.1755
Epoch 2/500
252/252 [==============================] - 0s 148us/sample - loss: 68.9950 -
mae: 7.4676 - mse: 68.9950 - val_loss: 60.1650 - val_mae: 7.0530 - val_mse:
60.1650
Epoch 3/500
252/252 [==============================] - 0s 163us/sample - loss: 35.7423 -
mae: 5.0151 - mse: 35.7423 - val_loss: 23.3648 - val_mae: 4.0795 - val_mse:
23.3648
Epoch 4/500
252/252 [==============================] - 0s 143us/sample - loss: 15.5794 -
mae: 3.1970 - mse: 15.5794 - val_loss: 12.1484 - val_mae: 2.7354 - val_mse:
12.1484
Epoch 5/500
252/252 [==============================] - 0s 139us/sample - loss: 16.7799 -
mae: 3.3424 - mse: 16.7799 - val_loss: 12.2341 - val_mae: 2.7190 - val_mse:
12.2341
Epoch 6/500
252/252 [==============================] - 0s 159us/sample - loss: 14.3608 -
mae: 3.0632 - mse: 14.3608 - val_loss: 13.1388 - val_mae: 3.0414 - val_mse:
13.1388
Epoch 7/500
252/252 [==============================] - 0s 139us/sample - loss: 13.4952 -
mae: 2.9374 - mse: 13.4952 - val_loss: 14.5711 - val_mae: 3.2084 - val_mse:
14.5711
Epoch 8/500
252/252 [==============================] - 0s 139us/sample - loss: 13.0871 -
mae: 2.8948 - mse: 13.0871 - val_loss: 12.8977 - val_mae: 2.9999 - val_mse:
12.8977
Epoch 9/500
252/252 [==============================] - 0s 171us/sample - loss: 12.4824 -
mae: 2.8223 - mse: 12.4824 - val_loss: 11.9152 - val_mae: 2.8388 - val_mse:
11.9152
Epoch 10/500
```

```
252/252 [==============================] - 0s 143us/sample - loss: 12.3469 -
mae: 2.8165 - mse: 12.3469 - val_loss: 11.9883 - val_mae: 2.8559 - val_mse:
11.9883
Epoch 11/500
252/252 [==============================] - 0s 147us/sample - loss: 11.9838 -
mae: 2.7674 - mse: 11.9838 - val_loss: 12.0808 - val_mae: 2.8677 - val_mse:
12.0808
Epoch 12/500
252/252 [==============================] - 0s 135us/sample - loss: 11.7020 -
mae: 2.7312 - mse: 11.7020 - val_loss: 12.1372 - val_mae: 2.8736 - val_mse:
12.1372
Epoch 13/500
252/252 [==============================] - 0s 171us/sample - loss: 11.4032 -
mae: 2.6896 - mse: 11.4032 - val_loss: 11.9152 - val_mae: 2.8319 - val_mse:
11.9152
Epoch 14/500
252/252 [==============================] - 0s 182us/sample - loss: 11.1822 -
mae: 2.6608 - mse: 11.1822 - val_loss: 11.9083 - val_mae: 2.8360 - val_mse:
11.9083
Epoch 15/500
252/252 [==============================] - 0s 178us/sample - loss: 10.9327 -
mae: 2.6308 - mse: 10.9327 - val_loss: 11.5259 - val_mae: 2.7675 - val_mse:
11.5259
Epoch 16/500
252/252 [==============================] - 0s 147us/sample - loss: 10.6598 -
mae: 2.5952 - mse: 10.6598 - val_loss: 11.5284 - val_mae: 2.7787 - val_mse:
11.5284
Epoch 17/500
252/252 [==============================] - 0s 194us/sample - loss: 10.4506 -
mae: 2.5689 - mse: 10.4506 - val_loss: 11.5317 - val_mae: 2.7945 - val_mse:
11.5317
Epoch 18/500
252/252 [==============================] - 0s 202us/sample - loss: 10.2737 -
mae: 2.5408 - mse: 10.2737 - val_loss: 11.6366 - val_mae: 2.8340 - val_mse:
11.6366
Epoch 19/500
252/252 [==============================] - 0s 190us/sample - loss: 9.9806 - mae:
2.5065 - mse: 9.9806 - val_loss: 11.1527 - val_mae: 2.7483 - val_mse: 11.1527
Epoch 20/500
252/252 [==============================] - 0s 186us/sample - loss: 9.8886 - mae:
2.5003 - mse: 9.8886 - val_loss: 10.8517 - val_mae: 2.6890 - val_mse: 10.8517
Epoch 21/500
252/252 [==============================] - 0s 190us/sample - loss: 9.5942 - mae:
2.4590 - mse: 9.5942 - val_loss: 11.3237 - val_mae: 2.8111 - val_mse: 11.3237
Epoch 22/500
252/252 [==============================] - 0s 178us/sample - loss: 9.5025 - mae:
2.4370 - mse: 9.5025 - val_loss: 11.1629 - val_mae: 2.7960 - val_mse: 11.1629
Epoch 23/500
```

```
252/252 [==============================] - 0s 206us/sample - loss: 9.2261 - mae:
2.4118 - mse: 9.2261 - val_loss: 10.5604 - val_mae: 2.6729 - val_mse: 10.5604
Epoch 24/500
252/252 [==============================] - 0s 178us/sample - loss: 9.1089 - mae:
2.3883 - mse: 9.1089 - val_loss: 10.9679 - val_mae: 2.7767 - val_mse: 10.9679
Epoch 25/500
252/252 [==============================] - 0s 159us/sample - loss: 8.8724 - mae:
2.3539 - mse: 8.8724 - val_loss: 10.7263 - val_mae: 2.7379 - val_mse: 10.7263
Epoch 26/500
252/252 [==============================] - 0s 171us/sample - loss: 8.8352 - mae:
2.3536 - mse: 8.8352 - val_loss: 10.5505 - val_mae: 2.7106 - val_mse: 10.5505
Epoch 27/500
252/252 [==============================] - 0s 159us/sample - loss: 8.5646 - mae:
2.3154 - mse: 8.5646 - val_loss: 10.6418 - val_mae: 2.7370 - val_mse: 10.6418
Epoch 28/500
252/252 [==============================] - 0s 151us/sample - loss: 8.3598 - mae:
2.2812 - mse: 8.3598 - val_loss: 10.2916 - val_mae: 2.6788 - val_mse: 10.2916
Epoch 29/500
252/252 [==============================] - 0s 143us/sample - loss: 8.2028 - mae:
2.2621 - mse: 8.2028 - val_loss: 10.1067 - val_mae: 2.6553 - val_mse: 10.1067
Epoch 30/500
252/252 [==============================] - 0s 178us/sample - loss: 8.0883 - mae:
2.2411 - mse: 8.0883 - val_loss: 10.4662 - val_mae: 2.7265 - val_mse: 10.4662
Epoch 31/500
252/252 [==============================] - 0s 139us/sample - loss: 7.8924 - mae:
2.2105 - mse: 7.8924 - val_loss: 9.9715 - val_mae: 2.6448 - val_mse: 9.9715
Epoch 32/500
252/252 [==============================] - 0s 171us/sample - loss: 7.7437 - mae:
2.1896 - mse: 7.7437 - val_loss: 10.3203 - val_mae: 2.7126 - val_mse: 10.3203
Epoch 33/500
252/252 [==============================] - 0s 163us/sample - loss: 7.6496 - mae:
2.1799 - mse: 7.6496 - val_loss: 9.8242 - val_mae: 2.6348 - val_mse: 9.8242
Epoch 34/500
252/252 [==============================] - 0s 186us/sample - loss: 7.4114 - mae:
2.1447 - mse: 7.4114 - val_loss: 10.1825 - val_mae: 2.6958 - val_mse: 10.1825
Epoch 35/500
252/252 [==============================] - 0s 190us/sample - loss: 7.2440 - mae:
2.1118 - mse: 7.2440 - val_loss: 9.9736 - val_mae: 2.6678 - val_mse: 9.9736
Epoch 36/500
252/252 [==============================] - 0s 163us/sample - loss: 7.0716 - mae:
2.0895 - mse: 7.0716 - val_loss: 9.9218 - val_mae: 2.6618 - val_mse: 9.9218
Epoch 37/500
252/252 [==============================] - 0s 139us/sample - loss: 6.9647 - mae:
2.0767 - mse: 6.9647 - val_loss: 9.7703 - val_mae: 2.6368 - val_mse: 9.7703
Epoch 38/500
252/252 [==============================] - 0s 131us/sample - loss: 6.8314 - mae:
2.0463 - mse: 6.8314 - val_loss: 9.8697 - val_mae: 2.6554 - val_mse: 9.8697
Epoch 39/500
```

```
252/252 [==============================] - 0s 147us/sample - loss: 6.5984 - mae:
2.0144 - mse: 6.5984 - val_loss: 9.6966 - val_mae: 2.6357 - val_mse: 9.6966
Epoch 40/500
252/252 [==============================] - 0s 135us/sample - loss: 6.4547 - mae:
1.9932 - mse: 6.4547 - val_loss: 9.4722 - val_mae: 2.6058 - val_mse: 9.4722
Epoch 41/500
252/252 [==============================] - 0s 123us/sample - loss: 6.2796 - mae:
1.9672 - mse: 6.2796 - val_loss: 9.7264 - val_mae: 2.6439 - val_mse: 9.7264
Epoch 42/500
252/252 [==============================] - 0s 127us/sample - loss: 6.1437 - mae:
1.9311 - mse: 6.1437 - val_loss: 9.5780 - val_mae: 2.6203 - val_mse: 9.5780
Epoch 43/500
252/252 [==============================] - 0s 127us/sample - loss: 6.0092 - mae:
1.9170 - mse: 6.0092 - val_loss: 9.5028 - val_mae: 2.6071 - val_mse: 9.5028
Epoch 44/500
252/252 [==============================] - 0s 127us/sample - loss: 5.8846 - mae:
1.8940 - mse: 5.8846 - val_loss: 9.5872 - val_mae: 2.6222 - val_mse: 9.5872
Epoch 45/500
252/252 [==============================] - 0s 123us/sample - loss: 5.5878 - mae:
1.8421 - mse: 5.5878 - val_loss: 9.5645 - val_mae: 2.6103 - val_mse: 9.5645
Epoch 46/500
252/252 [==============================] - 0s 135us/sample - loss: 5.3964 - mae:
1.8039 - mse: 5.3964 - val_loss: 9.6046 - val_mae: 2.6124 - val_mse: 9.6046
Epoch 47/500
252/252 [==============================] - 0s 139us/sample - loss: 5.3222 - mae:
1.7912 - mse: 5.3222 - val_loss: 9.6682 - val_mae: 2.6155 - val_mse: 9.6682
Epoch 48/500
252/252 [==============================] - 0s 131us/sample - loss: 5.1316 - mae:
1.7522 - mse: 5.1316 - val_loss: 9.6689 - val_mae: 2.6091 - val_mse: 9.6689
Epoch 49/500
252/252 [==============================] - 0s 111us/sample - loss: 5.1285 - mae:
1.7618 - mse: 5.1285 - val_loss: 9.3709 - val_mae: 2.5656 - val_mse: 9.3709
Epoch 50/500
252/252 [==============================] - 0s 123us/sample - loss: 4.8228 - mae:
1.6921 - mse: 4.8228 - val_loss: 9.7656 - val_mae: 2.6181 - val_mse: 9.7656
Epoch 51/500
252/252 [==============================] - 0s 143us/sample - loss: 4.6952 - mae:
1.6716 - mse: 4.6952 - val_loss: 9.6214 - val_mae: 2.5926 - val_mse: 9.6214
Epoch 52/500
252/252 [==============================] - 0s 131us/sample - loss: 4.4276 - mae:
1.6128 - mse: 4.4276 - val_loss: 9.8214 - val_mae: 2.6153 - val_mse: 9.8214
Epoch 53/500
252/252 [==============================] - 0s 123us/sample - loss: 4.4687 - mae:
1.6119 - mse: 4.4687 - val_loss: 9.5321 - val_mae: 2.5791 - val_mse: 9.5321
Epoch 54/500
252/252 [==============================] - 0s 127us/sample - loss: 4.2847 - mae:
1.6009 - mse: 4.2847 - val_loss: 9.6772 - val_mae: 2.5956 - val_mse: 9.6772
Epoch 55/500
```

```
252/252 [==============================] - 0s 127us/sample - loss: 4.0750 - mae:
1.5377 - mse: 4.0750 - val_loss: 9.7804 - val_mae: 2.5977 - val_mse: 9.7804
Epoch 56/500
252/252 [==============================] - 0s 119us/sample - loss: 3.7738 - mae:
1.4719 - mse: 3.7738 - val_loss: 9.5685 - val_mae: 2.5684 - val_mse: 9.5685
Epoch 57/500
252/252 [==============================] - 0s 127us/sample - loss: 3.6458 - mae:
1.4480 - mse: 3.6458 - val_loss: 9.8789 - val_mae: 2.5957 - val_mse: 9.8789
Epoch 58/500
252/252 [==============================] - 0s 131us/sample - loss: 3.4813 - mae:
1.4147 - mse: 3.4813 - val_loss: 9.7809 - val_mae: 2.6105 - val_mse: 9.7809
Epoch 59/500
252/252 [==============================] - 0s 135us/sample - loss: 3.2974 - mae:
1.3672 - mse: 3.2974 - val_loss: 9.9497 - val_mae: 2.6044 - val_mse: 9.9497
```

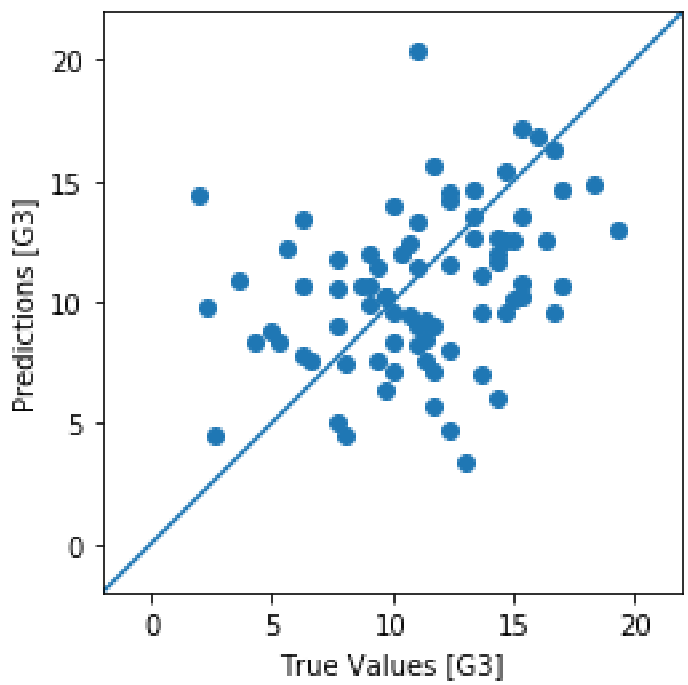
[32]:
```
<tensorflow.python.keras.callbacks.History at 0x1c56e426208>
```

[33]:
```
error = model.evaluate(x_test, y_test, verbose=2)
```

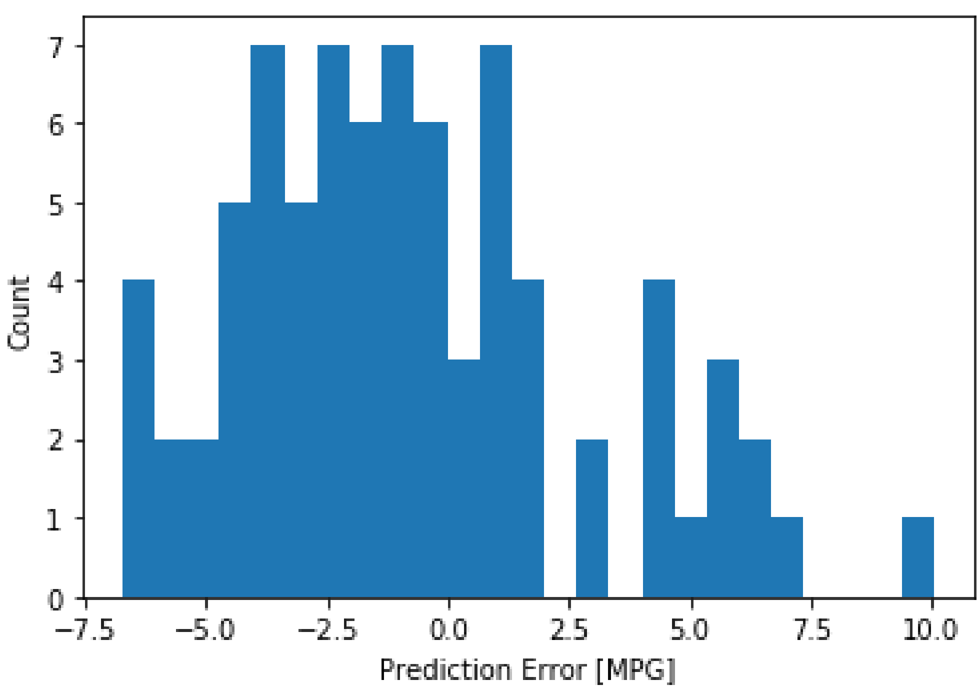
```
79/79 - 0s - loss: 13.8183 - mae: 2.9851 - mse: 13.8183
```

[26]:
```
prediction = model.predict(x_test).flatten()

a = plt.axes(aspect='equal')
plt.scatter(y_test, prediction)
plt.xlabel('True Values [G3]')
plt.ylabel('Predictions [G3]')
lims = [-2, 22]
plt.xlim(lims)
plt.ylim(lims)
_ = plt.plot(lims, lims)
```



[699]:
```
error = prediction - y_test
plt.hist(error, bins = 25)
plt.xlabel("Prediction Error [MPG]")
_ = plt.ylabel("Count")
```



[ ]:

```

```