spaceship-titanic

June 25, 2023

[1]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that␣
↪gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved␣
↪outside of the current session
```

```
/kaggle/input/spaceship-titanic/sample_submission.csv
/kaggle/input/spaceship-titanic/train.csv
/kaggle/input/spaceship-titanic/test.csv
```

[2]:
```
# Import data
train = pd.read_csv('/kaggle/input/spaceship-titanic/train.csv')
test = pd.read_csv('/kaggle/input/spaceship-titanic/test.csv')
train.head()
```

[2]:
```
  PassengerId HomePlanet CryoSleep  Cabin  Destination   Age    VIP  \
0     0001_01     Europa     False  B/0/P  TRAPPIST-1e  39.0  False
1     0002_01      Earth     False  F/0/S  TRAPPIST-1e  24.0  False
2     0003_01     Europa     False  A/0/S  TRAPPIST-1e  58.0   True
3     0003_02     Europa     False  A/0/S  TRAPPIST-1e  33.0  False
```

```
4     0004_01      Earth     False  F/1/S  TRAPPIST-1e  16.0  False

   RoomService  FoodCourt  ShoppingMall     Spa  VRDeck               Name  \
0          0.0        0.0           0.0     0.0     0.0    Maham Ofracculy
1        109.0        9.0          25.0   549.0    44.0       Juanna Vines
2         43.0     3576.0           0.0  6715.0    49.0      Altark Susent
3          0.0     1283.0         371.0  3329.0   193.0       Solam Susent
4        303.0       70.0         151.0   565.0     2.0  Willy Santantines

   Transported
0        False
1         True
2        False
3        False
4         True
```

[3]:
```
# Clean data
df = train.dropna(inplace=True)
df = train.drop('Name', axis=1)
df.head()
```

[3]:
```
  PassengerId HomePlanet CryoSleep  Cabin  Destination   Age    VIP  \
0     0001_01     Europa     False  B/0/P  TRAPPIST-1e  39.0  False
1     0002_01      Earth     False  F/0/S  TRAPPIST-1e  24.0  False
2     0003_01     Europa     False  A/0/S  TRAPPIST-1e  58.0   True
3     0003_02     Europa     False  A/0/S  TRAPPIST-1e  33.0  False
4     0004_01      Earth     False  F/1/S  TRAPPIST-1e  16.0  False

   RoomService  FoodCourt  ShoppingMall     Spa  VRDeck  Transported
0          0.0        0.0           0.0     0.0     0.0        False
1        109.0        9.0          25.0   549.0    44.0         True
2         43.0     3576.0           0.0  6715.0    49.0        False
3          0.0     1283.0         371.0  3329.0   193.0        False
4        303.0       70.0         151.0   565.0     2.0         True
```

[4]:
```
# Create list of column names and check data type
cols = list(df.columns)
dtypes = df.dtypes
print(dtypes)
```

```
PassengerId      object
HomePlanet       object
CryoSleep        object
Cabin            object
Destination      object
Age             float64
VIP              object
```

```
RoomService     float64
FoodCourt       float64
ShoppingMall    float64
Spa             float64
VRDeck          float64
Transported        bool
dtype: object
```

[5]:
```
# Turn passenger ID into float
def pid_to_float(d):
    ks = []
    vs = []
    for ix, row in d.iterrows():
        val = row['PassengerId']
        ks.append(val)
        nval = val.replace('_', '')
        vs.append(nval)
    kvar = np.asarray(vs, dtype=np.float64)
    passid = {ks[i]: kvar[i] for i in range(len(ks))}
    return passid

passid = pid_to_float(df)
df['PassengerId'] = df['PassengerId'].map(passid)
```

[6]:
```
# Non-float columns into floats
nf_cols = []
for col in cols:
    if df[col].dtype != np.float64:
        nf_cols.append(col)

def sb_to_float(d, n):
    vdl = []
    for c in nf_cols:
        col = d[c]
        vals = list(col[0:])
        v_dict = {}
        i = float(n)
        for v in vals:
            if v in list(v_dict.keys()):
                continue
            else:
                v_dict[v] = i
                i += 1
        vdl.append(v_dict)
    return vdl

vdl = sb_to_float(df, 0)
```

```
print(vdl[1])
```

```
{False: 0.0, True: 1.0}
```

[7]:
```
# Change values in columns
for i in range(len(vdl)):
    cl = nf_cols[i]
    df[cl] = df[cl].map(vdl[i])

df.head
```

[7]:
```
<bound method NDFrame.head of       PassengerId  HomePlanet  CryoSleep   Cabin
Destination   Age  VIP  \
0           101.0         0.0        0.0     0.0          0.0  39.0  0.0
1           201.0         1.0        0.0     1.0          0.0  24.0  0.0
2           301.0         0.0        0.0     2.0          0.0  58.0  1.0
3           302.0         0.0        0.0     2.0          0.0  33.0  0.0
4           401.0         1.0        0.0     3.0          0.0  16.0  0.0
…           …           …        …     …           …    …   …
8688     927601.0         0.0        0.0  5301.0          2.0  41.0  1.0
8689     927801.0         1.0        1.0  5302.0          1.0  18.0  0.0
8690     927901.0         1.0        0.0  5303.0          0.0  26.0  0.0
8691     928001.0         0.0        0.0  5304.0          2.0  32.0  0.0
8692     928002.0         0.0        0.0  5304.0          0.0  44.0  0.0

      RoomService  FoodCourt  ShoppingMall     Spa  VRDeck  Transported
0             0.0        0.0           0.0     0.0     0.0          0.0
1           109.0        9.0          25.0   549.0    44.0          1.0
2            43.0     3576.0           0.0  6715.0    49.0          0.0
3             0.0     1283.0         371.0  3329.0   193.0          0.0
4           303.0       70.0         151.0   565.0     2.0          1.0
…           …         …           …     …     …       …
8688          0.0     6819.0           0.0  1643.0    74.0          0.0
8689          0.0        0.0           0.0     0.0     0.0          0.0
8690          0.0        0.0        1872.0     1.0     0.0          1.0
8691          0.0     1049.0           0.0   353.0  3235.0          0.0
8692        126.0     4688.0           0.0     0.0    12.0          1.0

[6606 rows x 13 columns]>
```

[8]:
```
# Split data
x_train = df.drop('Transported', axis=1)
y_train = df['Transported']
```

[9]:
```
# Normalize data
import tensorflow as tf
```

```
norm = tf.keras.layers.Normalization()
norm.adapt(x_train)
```

[10]:
```
# Model
def build_and_compile_model(norm):
  model = tf.keras.Sequential([
      norm,
      tf.keras.layers.Dense(64, activation='relu'),
      tf.keras.layers.Dense(64, activation='sigmoid'),
      tf.keras.layers.Dense(32, activation='relu'),
      tf.keras.layers.Dense(16, activation='sigmoid'),
      tf.keras.layers.Dense(16, activation='relu'),
      tf.keras.layers.Dense(8, activation='sigmoid'),
      tf.keras.layers.Dense(1)
  ])

  model.compile(loss='mean_absolute_error',
                optimizer=tf.keras.optimizers.Adamax(0.0055))
  return model

linear_model = build_and_compile_model(norm)

history = linear_model.fit(
    x_train,
    y_train,
    epochs=100,
    validation_split = 0.085)
```

```
Epoch 1/100
189/189 [==============================] - 2s 4ms/step - loss: 0.3721 -
val_loss: 0.2141
Epoch 2/100
189/189 [==============================] - 0s 2ms/step - loss: 0.2177 -
val_loss: 0.2308
Epoch 3/100
189/189 [==============================] - 0s 3ms/step - loss: 0.2106 -
val_loss: 0.2084
Epoch 4/100
189/189 [==============================] - 0s 2ms/step - loss: 0.2065 -
val_loss: 0.2135
Epoch 5/100
189/189 [==============================] - 0s 2ms/step - loss: 0.2074 -
val_loss: 0.2084
Epoch 6/100
189/189 [==============================] - 0s 2ms/step - loss: 0.2041 -
val_loss: 0.1972
Epoch 7/100
```

```
189/189 [==============================] - 0s 2ms/step - loss: 0.2035 -
val_loss: 0.2062
Epoch 8/100
189/189 [==============================] - 0s 3ms/step - loss: 0.2003 -
val_loss: 0.2001
Epoch 9/100
189/189 [==============================] - 0s 2ms/step - loss: 0.2019 -
val_loss: 0.1989
Epoch 10/100
189/189 [==============================] - 0s 3ms/step - loss: 0.2001 -
val_loss: 0.2030
Epoch 11/100
189/189 [==============================] - 0s 2ms/step - loss: 0.2001 -
val_loss: 0.1993
Epoch 12/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1983 -
val_loss: 0.1983
Epoch 13/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1961 -
val_loss: 0.1960
Epoch 14/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1981 -
val_loss: 0.2050
Epoch 15/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1983 -
val_loss: 0.2289
Epoch 16/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1985 -
val_loss: 0.2172
Epoch 17/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1970 -
val_loss: 0.2038
Epoch 18/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1952 -
val_loss: 0.2092
Epoch 19/100
189/189 [==============================] - 1s 3ms/step - loss: 0.1968 -
val_loss: 0.1996
Epoch 20/100
189/189 [==============================] - 0s 3ms/step - loss: 0.1957 -
val_loss: 0.2065
Epoch 21/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1981 -
val_loss: 0.2038
Epoch 22/100
189/189 [==============================] - 1s 3ms/step - loss: 0.1988 -
val_loss: 0.2029
Epoch 23/100
```

```
189/189 [==============================] - 0s 2ms/step - loss: 0.1971 -
val_loss: 0.2060
Epoch 24/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1988 -
val_loss: 0.2056
Epoch 25/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1975 -
val_loss: 0.2181
Epoch 26/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1950 -
val_loss: 0.1996
Epoch 27/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1949 -
val_loss: 0.2003
Epoch 28/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1956 -
val_loss: 0.2019
Epoch 29/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1968 -
val_loss: 0.2027
Epoch 30/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1962 -
val_loss: 0.2107
Epoch 31/100
189/189 [==============================] - 0s 3ms/step - loss: 0.1946 -
val_loss: 0.1977
Epoch 32/100
189/189 [==============================] - 1s 3ms/step - loss: 0.1969 -
val_loss: 0.2009
Epoch 33/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1952 -
val_loss: 0.1996
Epoch 34/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1925 -
val_loss: 0.2115
Epoch 35/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1929 -
val_loss: 0.2031
Epoch 36/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1979 -
val_loss: 0.1983
Epoch 37/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1963 -
val_loss: 0.2113
Epoch 38/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1927 -
val_loss: 0.2012
Epoch 39/100
```

```
189/189 [==============================] - 0s 2ms/step - loss: 0.1909 -
val_loss: 0.1986
Epoch 40/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1912 -
val_loss: 0.2013
Epoch 41/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1941 -
val_loss: 0.1991
Epoch 42/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1937 -
val_loss: 0.1970
Epoch 43/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1934 -
val_loss: 0.2029
Epoch 44/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1925 -
val_loss: 0.2055
Epoch 45/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1904 -
val_loss: 0.1975
Epoch 46/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1907 -
val_loss: 0.1979
Epoch 47/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1921 -
val_loss: 0.2028
Epoch 48/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1921 -
val_loss: 0.2009
Epoch 49/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1904 -
val_loss: 0.1980
Epoch 50/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1910 -
val_loss: 0.2184
Epoch 51/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1943 -
val_loss: 0.2152
Epoch 52/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1900 -
val_loss: 0.2087
Epoch 53/100
189/189 [==============================] - 0s 3ms/step - loss: 0.1952 -
val_loss: 0.2074
Epoch 54/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1917 -
val_loss: 0.2002
Epoch 55/100
```

```
189/189 [==============================] - 0s 2ms/step - loss: 0.1904 -
val_loss: 0.2025
Epoch 56/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1901 -
val_loss: 0.2126
Epoch 57/100
189/189 [==============================] - 0s 3ms/step - loss: 0.1901 -
val_loss: 0.1987
Epoch 58/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1899 -
val_loss: 0.1994
Epoch 59/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1877 -
val_loss: 0.1995
Epoch 60/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1913 -
val_loss: 0.2106
Epoch 61/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1905 -
val_loss: 0.2089
Epoch 62/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1909 -
val_loss: 0.1962
Epoch 63/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1963 -
val_loss: 0.1946
Epoch 64/100
189/189 [==============================] - 0s 3ms/step - loss: 0.1916 -
val_loss: 0.1978
Epoch 65/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1899 -
val_loss: 0.1999
Epoch 66/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1888 -
val_loss: 0.2009
Epoch 67/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1911 -
val_loss: 0.1958
Epoch 68/100
189/189 [==============================] - 0s 3ms/step - loss: 0.1886 -
val_loss: 0.2010
Epoch 69/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1889 -
val_loss: 0.2140
Epoch 70/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1898 -
val_loss: 0.1998
Epoch 71/100
```

```
189/189 [==============================] - 0s 2ms/step - loss: 0.1889 -
val_loss: 0.2012
Epoch 72/100
189/189 [==============================] - 0s 3ms/step - loss: 0.1872 -
val_loss: 0.2014
Epoch 73/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1875 -
val_loss: 0.1979
Epoch 74/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1878 -
val_loss: 0.1976
Epoch 75/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1898 -
val_loss: 0.2069
Epoch 76/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1962 -
val_loss: 0.2097
Epoch 77/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1904 -
val_loss: 0.1944
Epoch 78/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1886 -
val_loss: 0.2092
Epoch 79/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1917 -
val_loss: 0.2037
Epoch 80/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1882 -
val_loss: 0.2006
Epoch 81/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1880 -
val_loss: 0.1985
Epoch 82/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1894 -
val_loss: 0.1956
Epoch 83/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1876 -
val_loss: 0.2018
Epoch 84/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1903 -
val_loss: 0.2089
Epoch 85/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1881 -
val_loss: 0.1996
Epoch 86/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1869 -
val_loss: 0.2008
Epoch 87/100
```

```
189/189 [==============================] - 0s 2ms/step - loss: 0.1873 -
val_loss: 0.1978
Epoch 88/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1876 -
val_loss: 0.2013
Epoch 89/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1894 -
val_loss: 0.2124
Epoch 90/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1885 -
val_loss: 0.1978
Epoch 91/100
189/189 [==============================] - 1s 3ms/step - loss: 0.1882 -
val_loss: 0.2007
Epoch 92/100
189/189 [==============================] - 1s 3ms/step - loss: 0.1864 -
val_loss: 0.1949
Epoch 93/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1867 -
val_loss: 0.2123
Epoch 94/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1897 -
val_loss: 0.1971
Epoch 95/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1890 -
val_loss: 0.2026
Epoch 96/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1890 -
val_loss: 0.1953
Epoch 97/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1878 -
val_loss: 0.2018
Epoch 98/100
189/189 [==============================] - 0s 2ms/step - loss: 0.1863 -
val_loss: 0.2030
Epoch 99/100
189/189 [==============================] - 0s 3ms/step - loss: 0.1884 -
val_loss: 0.1971
Epoch 100/100
189/189 [==============================] - 0s 3ms/step - loss: 0.1879 -
val_loss: 0.2002
```

[11]:

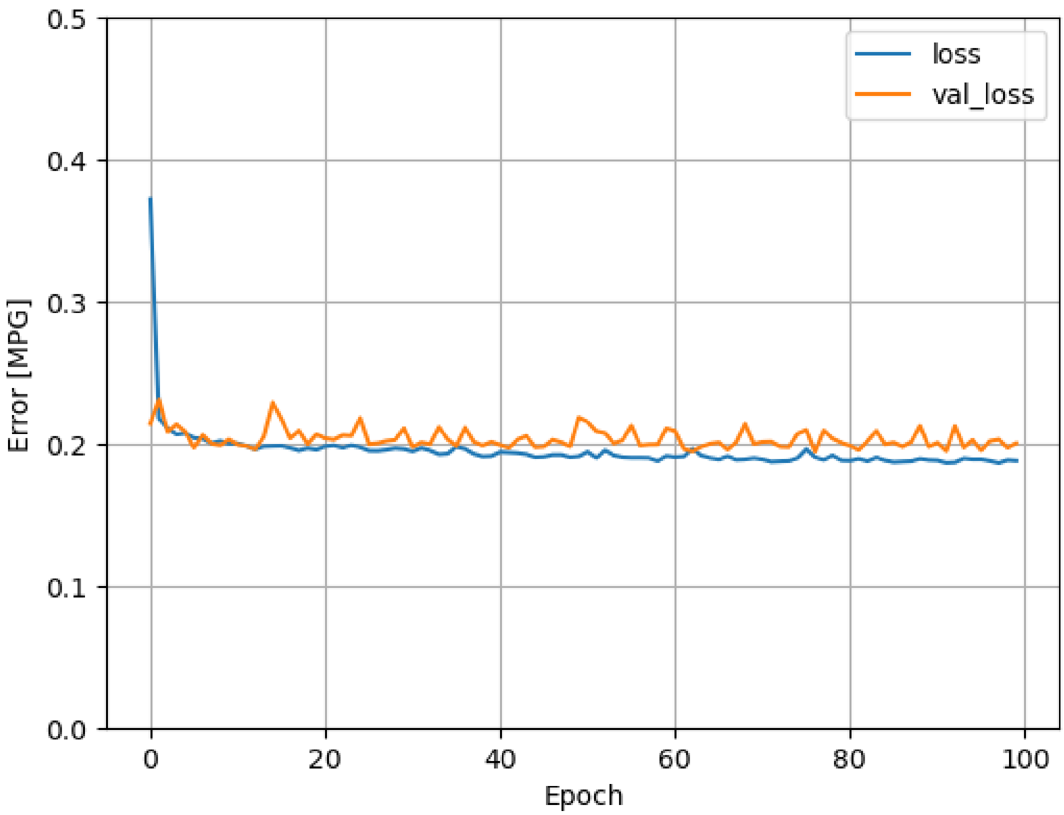
```
# Plot data
import matplotlib.pyplot as plt

hist = pd.DataFrame(history.history)
hist['epoch'] = history.epoch
```

```
hist.tail()

def plot_loss(history):
  plt.plot(history.history['loss'], label='loss')
  plt.plot(history.history['val_loss'], label='val_loss')
  plt.ylim([0, 0.5])
  plt.xlabel('Epoch')
  plt.ylabel('Error [MPG]')
  plt.legend()
  plt.grid(True)

plot_loss(history)
```



[12]:
```
# Test results
test_results = {}

test_results['linear_model'] = linear_model.evaluate(
    x_train,
    y_train, verbose=0)

print(test_results)
```

```
{'linear_model': 0.18560750782489777}
```

[13]:
```
# Convert test data
t = test
df2 = t.drop('Name', axis=1)

pid2 = pid_to_float(test)
cpid = df2['PassengerId']
df2['PassengerId'] = df2['PassengerId'].map(pid2)

def conv_df2():
    v2l = []
    for c in nf_cols:
        if c != 'Transported':
            col = df2[c]
            n = len(col) + 1
            vl = list(col)
            v2k = []
            v2v = []
            for val in col:
                v2k.append(val)
                v2v.append(n)
                n += 1
            v2vf = np.array(v2v, dtype=np.float64)
            v2 = {v2k[i]: v2vf[i] for i in range(len(v2k))}
            v2l.append(v2)
    return v2l

cv2 = conv_df2()

def merge_two_dicts(d1, d2):
    d3 = d1.copy()
    for key, value in d2.items():
        if key not in list(d1.keys()):
            d3[key] = value
    return d3

md = []
for i in range(len(cv2)):
    m2 = merge_two_dicts(vdl[i], cv2[i])
    md.append(m2)

for n in range(len(cv2)):
    col = nf_cols[n]
    for val in df2[col]:
        i = 0
        if val in list(md[n].keys()):
```

```
            nv = md[n].get(val)
            df2 = df2.replace(to_replace=val, value=nv)
```

[14]:

```
# Normalize test data
norm.adapt(df2)
print(norm(df2))
```

```
tf.Tensor(
[[-1.703426   -0.14408089 -0.14857945 … -0.27006406 -0.2985787
  -0.2794406 ]
 [-1.7015849  -0.14408089 -0.14938699 … -0.27006406  1.3993948
  -0.2794406 ]
 [-1.7012167  -0.14491527 -0.14857945 … -0.27006406 -0.2985787
  -0.2794406 ]
 …
 [ 1.7054143  -0.14324652 -0.14857945 … -0.27006406 -0.2985787
  -0.2794406 ]
 [ 1.7061507  -0.14491527 -0.14938699 … -0.27006406 -0.2985787
   0.03572261]
 [ 1.7076235  -0.14408089 -0.14857945 … -0.27006406 -0.2985787
  -0.2794406 ]], shape=(4277, 12), dtype=float32)
```

[15]:

```
# Make predictions
predictions = linear_model.predict(df2)
pcol = []
for p in predictions:
    pl = round(p[0])
    pcol.append(pl)
len(pcol)
```

```
134/134 [==============================] - 0s 2ms/step
```

[15]: 4277

[16]:

```
# Create and save submission dataframe
Transported = pd.Series(pcol)
pcdc = {0: False, 1: True}
submit = pd.concat([cpid, Transported], axis=1)
submit = submit[:len(pcol)]
submit.columns = [submit.columns[0], 'Transported']
submit['Transported'] = submit['Transported'].map(pcdc)
submit.to_csv('/kaggle/working/submission.csv', index=False)
submit.head()
```

[16]:

	PassengerId	Transported
0	0013_01	True
1	0018_01	False

```
2     0019_01          True
3     0021_01          True
4     0023_01          True
```

```
[ ]:
```