IC50_Prediction_Baseline_Submission

May 13, 2025

1 Simple feed forward NN to predict IC50 based on patient demographic and molecular drug information (BASELINE MODEL)

[32]:
```
from CHEM277B_functions import *
```

[33]:
```
valid_IC50s = pd.read_csv("data/valid_IC50s_within_range.csv")
merged_df = pd.read_csv("data/final_merged.csv")
lucas_df = pd.read_csv("data/final_mapping.csv")
cell_lines_df = pd.read_csv("data/HarvardCellLines.csv")
```

```
/var/folders/s8/ghqk1l4n7n9_w17t7hx21g2m0000gn/T/ipykernel_14517/2977721343.py:2
: DtypeWarning: Columns (0,3,4,5,6,7,9,12,15,16,17,18,22,24,26,28,29,30,40,65,66
,67,68,69,70,71,72,73,74,75,76,82,83,89,90,91,92,93,96,97,98,99,100,101,102,103,
104,105,106,107,108,109,115) have mixed types. Specify dtype option on import or
set low_memory=False.
  merged_df = pd.read_csv("final_merged.csv")
```

[34]:
```
valid_IC50s.drop(columns = ['Unnamed: 0', 'N Points'], inplace = True)
```

[35]:
```
columns = ["HMS LINCS ID", "Name", "T Stage"]
cell_lines_df = cell_lines_df[columns]
```

[36]:
```
merged_df["Race"].value_counts()
```

[36]:
```
Race
1.0    5857
2.0     470
4.0     321
0.0      19
5.0      18
3.0      14
6.0       9
8.0       4
7.0       1
Name: count, dtype: int64
```

[37]:
```
# TEMPORARY IMPUTATION OF RACE BASED OFF PROPORTIONS OF EXISTING RACES.
# WILL USE AUSTIN'S IMPUTED RACE ANALYSIS LATER.

# Get value counts as probabilities
race_dist = merged_df['Race'].value_counts(normalize=True)

# Get the indices where race is missing
missing_indices = merged_df['Race'].isna()

# Sample values based on observed distribution
imputed_values = np.random.choice(race_dist.index, size=missing_indices.sum(),␣
↪p=race_dist.values)

# Assign the sampled values to the missing positions
merged_df.loc[missing_indices, 'Race'] = imputed_values
```

[38]:
```
merged_df['T_stage_by_size'] = merged_df.apply(lambda row: row['T Stage'] if pd.
↪notnull(row['T Stage']) else T_stage_by_size(row['Tumor Size']), axis=1)
```

[39]:
```
columns = ['Age', 'Race', 'T_stage_by_size']
patients_df = merged_df[columns]
```

[40]:
```
def get_biased_cell_lines(patient_row, cell_line_df, n=3):
    # Filter for matching T_stage
    filtered = cell_line_df[cell_line_df["T Stage"] ==␣
↪patient_row["T_stage_by_size"]]

    # If fewer than n matches, fall back to all cell lines
    if len(filtered) < n:
        filtered = cell_line_df

    return np.random.choice(filtered["HMS LINCS ID"], size=n, replace=False).
↪tolist()
```

[41]:
```
patients_df["T_stage_by_size"].apply(type).value_counts()
```

[41]:
```
T_stage_by_size
<class 'float'>    9222
Name: count, dtype: int64
```

[42]:
```
cell_lines_df["T Stage"].apply(type).value_counts()
```

[42]:
```
T Stage
<class 'float'>    35
Name: count, dtype: int64
```

[43]:
```
# Add a list of 3 biased cell lines per patient
patients_df["cell_lines"] = patients_df.apply(lambda row:␣
↪get_biased_cell_lines(row, cell_lines_df), axis=1)
```

```
/var/folders/s8/ghqk1l4n7n9_w17t7hx21g2m0000gn/T/ipykernel_14517/1392642400.py:2
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  patients_df["cell_lines"] = patients_df.apply(lambda row:
get_biased_cell_lines(row, cell_lines_df), axis=1)
```

[44]:
```
patients_df
```

[44]:

	Age	Race	T_stage_by_size	cell_lines
0	41	2.0	2.0	[50205, 50216, 50211]
1	41	2.0	2.0	[50579, 50211, 50205]
2	38	2.0	2.0	[50211, 50212, 50205]
3	62	1.0	2.0	[50216, 50213, 50205]
4	62	1.0	2.0	[50205, 50579, 50211]
…	…	…	…	…
9217	69	1.0	4.0	[50335, 51083, 50057]
9218	69	1.0	4.0	[50331, 50105, 51081]
9219	69	1.0	4.0	[50327, 50331, 50334]
9220	69	1.0	4.0	[50008, 50328, 50335]
9221	69	1.0	4.0	[50238, 50029, 50208]

[9222 rows x 4 columns]

[45]:
```
# Explode so each row = 1 (patient, cell_line)
patients_df = patients_df.explode("cell_lines").reset_index(drop=True)
```

[46]:
```
cell_lines_df_dict = cell_lines_df.set_index("HMS LINCS ID").to_dict()["Name"]
```

[47]:
```
patients_df["Cell Name"] = patients_df["cell_lines"].apply(mapping, dictionary␣
↪= cell_lines_df_dict)
```

[48]:
```
patients_df
```

[48]:

	Age	Race	T_stage_by_size	cell_lines	Cell Name
0	41	2.0	2.0	50205	HCC1143
1	41	2.0	2.0	50216	HCC38
2	41	2.0	2.0	50211	HCC1806
3	41	2.0	2.0	50579	HCC1500
4	41	2.0	2.0	50211	HCC1806

```
...     ...  ...             ...         ...         ...
27661   69   1.0                 4.0       50328  MDA-MB-157
27662   69   1.0                 4.0       50335  MDA-MB-468
27663   69   1.0                 4.0       50238     Hs 578T
27664   69   1.0                 4.0       50029        MCF7
27665   69   1.0                 4.0       50208     HCC1428

[27666 rows x 5 columns]
```

[49]:
```
patients_df.dropna(inplace=True)
patients_df.isna().sum()
```

[49]:
```
Age                 0
Race                0
T_stage_by_size     0
cell_lines          0
Cell Name           0
dtype: int64
```

2 Add Small Molecule descriptors for each row.

[51]:
```
drugs_df = pd.read_csv("Descriptors_Small_Molecules.csv")
drugs_df = drugs_df[['Name', 'Molecular Mass', 'LogP', 'NumHDonors',␣
 ↪'NumHAcceptors', 'TPSA']]
drugs_df.rename(columns={'Name': 'Small Molecule Name'}, inplace=True)
drugs_df.head()
```

[51]:
```
  Small Molecule Name  Molecular Mass     LogP  NumHDonors  NumHAcceptors  \
0            AZD7762          362.12  2.52660           4              4
1          Neratinib          556.20  5.93248           2              8
2          Dasatinib          487.16  3.31354           3              9
3        Saracatinib          541.21  3.93950           1             10
4         Pictilisib          513.16  2.14840           1              9

     TPSA
0   96.25
1  112.40
2  106.51
3   90.44
4  107.55
```

[52]:
```
valid_IC50s.columns
```

[52]:
```
Index(['Cell Name', 'Small Molecule Name', 'EC50 (uM)'], dtype='object')
```

[53]:
```
valid_IC50s = pd.merge(valid_IC50s, drugs_df, on='Small Molecule Name',␣
    ↪how='left')
```

[54]:
```
patient_drug_df = pd.merge(patients_df, valid_IC50s, on='Cell Name', how='left')
```

[55]:
```
patient_drug_df.drop(columns = "cell_lines", inplace=True)
```

[56]:
```
patient_drug_df.to_csv("patient_drug_information_aka_final_final_final.csv")
```

[57]:
```
race_dict = {0:'N/A', 1:"white", 2:"black", 3:"asian", 4:"native", 5:
    ↪"hispanic", 6:"multi", 7:"hawa", 8:"amer indian"}
```

[58]:
```
patient_drug_df["Race"] = patient_drug_df["Race"].apply(mapping, dictionary =␣
    ↪race_dict)
```

2.0.1 Cell lines are proxy for patients, so drop the cell lines column.

[60]:
```
patient_drug_df.drop(columns="Cell Name", inplace=True)
```

2.0.2 Need to one-hot-encode categorical featurs

[62]:
```
categorical_cols = ['Race', 'T_stage_by_size', 'Small Molecule Name']
patient_drug_df[categorical_cols] = patient_drug_df[categorical_cols].
    ↪astype('category')
```

[63]:
```
patient_drug_df_encoded = pd.get_dummies(patient_drug_df,␣
    ↪columns=categorical_cols, drop_first=True)
print(patient_drug_df_encoded.dtypes)
```

```
Age                                  int64
EC50 (uM)                          float64
Molecular Mass                     float64
LogP                               float64
NumHDonors                           int64
NumHAcceptors                        int64
TPSA                               float64
Race_amer indian                      bool
Race_asian                            bool
Race_black                            bool
Race_hawa                             bool
Race_hispanic                         bool
Race_multi                            bool
Race_native                           bool
Race_white                            bool
T_stage_by_size_1.0                   bool
T_stage_by_size_2.0                   bool
```

```
T_stage_by_size_3.0                    bool
T_stage_by_size_4.0                    bool
Small Molecule Name_ABT-737            bool
Small Molecule Name_AZD7762            bool
Small Molecule Name_Abemaciclib        bool
Small Molecule Name_Alpelisib          bool
Small Molecule Name_Bleomycin          bool
Small Molecule Name_Buparlisib         bool
Small Molecule Name_Cabozantinib       bool
Small Molecule Name_Cediranib          bool
Small Molecule Name_Ceritinib          bool
Small Molecule Name_Cisplatin          bool
Small Molecule Name_Dasatinib          bool
Small Molecule Name_Dinaciclib         bool
Small Molecule Name_Doxorubicin        bool
Small Molecule Name_Etoposide          bool
Small Molecule Name_Everolimus         bool
Small Molecule Name_INK-128            bool
Small Molecule Name_Ipatasertib        bool
Small Molecule Name_Luminespib         bool
Small Molecule Name_Neratinib          bool
Small Molecule Name_Olaparib           bool
Small Molecule Name_PF-4708671         bool
Small Molecule Name_Palbociclib        bool
Small Molecule Name_Pictilisib         bool
Small Molecule Name_Saracatinib        bool
Small Molecule Name_TGX221             bool
Small Molecule Name_Taselisib          bool
Small Molecule Name_Taxol              bool
Small Molecule Name_Tivantinib         bool
Small Molecule Name_Topotecan          bool
Small Molecule Name_Torin2             bool
Small Molecule Name_Trametinib         bool
Small Molecule Name_Volasertib         bool
Small Molecule Name_Vorinostat         bool
dtype: object
```

[64]:
```
bool_cols = patient_drug_df_encoded.select_dtypes(include='bool').columns
patient_drug_df_encoded[bool_cols] = patient_drug_df_encoded[bool_cols].
  ↪astype(int)
```

2.0.3 PRE-PROCESSING DONE. ONTO MODEL TRAINING.

[66]:
```
# Separate features (X) and target (y)
X = patient_drug_df_encoded.drop(columns=['EC50 (uM)'])  # Drop EC50 and␣
  ↪non-features
y = patient_drug_df_encoded['EC50 (uM)']  # EC50 is the target
```

```
# Standardize numerical features
scaler = StandardScaler()
X[['Age', 'Molecular Mass', 'LogP', 'NumHDonors', 'NumHAcceptors', 'TPSA']] =␣
↪scaler.fit_transform(X[['Age', 'Molecular Mass', 'LogP', 'NumHDonors',␣
↪'NumHAcceptors', 'TPSA']])
```

[67]:
```
# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[68]:
```
# Data
# Select drug descriptor columns
drug_feature_cols = ['Molecular Mass', 'LogP', 'NumHDonors', 'NumHAcceptors',␣
↪'TPSA']
drug_features = X_train[drug_feature_cols].to_numpy()  # (n_samples, 5)

# Drop drug features to get patient features
patient_features = X_train.drop(columns=drug_feature_cols).to_numpy()

# Convert to tensors
drug_tensor = torch.tensor(drug_features).float()
patient_tensor = torch.tensor(patient_features).float()
ic50_tensor = torch.tensor(y_train.to_numpy()).float()

model = DrugResponsePredictor(drug_input_dim=5,␣
↪patient_input_dim=patient_tensor.shape[1])
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)
loss_fn = nn.MSELoss()


# Training loop
model.train()
start_time = time.time()

my_epochs = []
my_losses = []

for epoch in range(300):  # increase epochs for real training
    optimizer.zero_grad()
    preds = model(drug_tensor, patient_tensor)
    loss = loss_fn(preds, ic50_tensor)
    loss.backward()
    optimizer.step()
    end_time = time.time()
    elapsed = end_time - start_time
    print(f"Epoch {epoch} | Loss: {loss.item():.4f}")
```

```python
    print(f"Time elapsed: {elapsed:.2f} seconds")
    my_epochs.append(epoch)
    my_losses.append(loss.item())
```

```
Epoch 0 | Loss: 4.5900
Time elapsed: 0.17 seconds
Epoch 1 | Loss: 4.5277
Time elapsed: 0.27 seconds
Epoch 2 | Loss: 4.4675
Time elapsed: 0.37 seconds
Epoch 3 | Loss: 4.4096
Time elapsed: 0.47 seconds
Epoch 4 | Loss: 4.3538
Time elapsed: 0.58 seconds
Epoch 5 | Loss: 4.3000
Time elapsed: 0.68 seconds
Epoch 6 | Loss: 4.2479
Time elapsed: 0.78 seconds
Epoch 7 | Loss: 4.1972
Time elapsed: 0.88 seconds
Epoch 8 | Loss: 4.1466
Time elapsed: 0.99 seconds
Epoch 9 | Loss: 4.0955
Time elapsed: 1.10 seconds
Epoch 10 | Loss: 4.0437
Time elapsed: 1.23 seconds
Epoch 11 | Loss: 3.9912
Time elapsed: 1.34 seconds
Epoch 12 | Loss: 3.9380
Time elapsed: 1.45 seconds
Epoch 13 | Loss: 3.8838
Time elapsed: 1.55 seconds
Epoch 14 | Loss: 3.8289
Time elapsed: 1.65 seconds
Epoch 15 | Loss: 3.7727
Time elapsed: 1.75 seconds
Epoch 16 | Loss: 3.7156
Time elapsed: 1.85 seconds
Epoch 17 | Loss: 3.6581
Time elapsed: 1.95 seconds
Epoch 18 | Loss: 3.5994
Time elapsed: 2.05 seconds
Epoch 19 | Loss: 3.5407
Time elapsed: 2.15 seconds
Epoch 20 | Loss: 3.4824
Time elapsed: 2.24 seconds
Epoch 21 | Loss: 3.4247
```

```
Time elapsed: 2.34 seconds
Epoch 22 | Loss: 3.3681
Time elapsed: 2.44 seconds
Epoch 23 | Loss: 3.3130
Time elapsed: 2.54 seconds
Epoch 24 | Loss: 3.2598
Time elapsed: 2.64 seconds
Epoch 25 | Loss: 3.2090
Time elapsed: 2.74 seconds
Epoch 26 | Loss: 3.1611
Time elapsed: 2.85 seconds
Epoch 27 | Loss: 3.1164
Time elapsed: 2.95 seconds
Epoch 28 | Loss: 3.0755
Time elapsed: 3.05 seconds
Epoch 29 | Loss: 3.0388
Time elapsed: 3.15 seconds
Epoch 30 | Loss: 3.0064
Time elapsed: 3.25 seconds
Epoch 31 | Loss: 2.9788
Time elapsed: 3.35 seconds
Epoch 32 | Loss: 2.9561
Time elapsed: 3.45 seconds
Epoch 33 | Loss: 2.9382
Time elapsed: 3.55 seconds
Epoch 34 | Loss: 2.9248
Time elapsed: 3.65 seconds
Epoch 35 | Loss: 2.9151
Time elapsed: 3.76 seconds
Epoch 36 | Loss: 2.9082
Time elapsed: 3.86 seconds
Epoch 37 | Loss: 2.9030
Time elapsed: 3.96 seconds
Epoch 38 | Loss: 2.8984
Time elapsed: 4.06 seconds
Epoch 39 | Loss: 2.8935
Time elapsed: 4.16 seconds
Epoch 40 | Loss: 2.8873
Time elapsed: 4.26 seconds
Epoch 41 | Loss: 2.8795
Time elapsed: 4.36 seconds
Epoch 42 | Loss: 2.8699
Time elapsed: 4.46 seconds
Epoch 43 | Loss: 2.8587
Time elapsed: 4.56 seconds
Epoch 44 | Loss: 2.8464
Time elapsed: 4.66 seconds
Epoch 45 | Loss: 2.8334
```

```
Time elapsed: 4.76 seconds
Epoch 46 | Loss: 2.8203
Time elapsed: 4.86 seconds
Epoch 47 | Loss: 2.8076
Time elapsed: 4.96 seconds
Epoch 48 | Loss: 2.7955
Time elapsed: 5.06 seconds
Epoch 49 | Loss: 2.7842
Time elapsed: 5.16 seconds
Epoch 50 | Loss: 2.7737
Time elapsed: 5.26 seconds
Epoch 51 | Loss: 2.7640
Time elapsed: 5.36 seconds
Epoch 52 | Loss: 2.7549
Time elapsed: 5.46 seconds
Epoch 53 | Loss: 2.7462
Time elapsed: 5.56 seconds
Epoch 54 | Loss: 2.7378
Time elapsed: 5.66 seconds
Epoch 55 | Loss: 2.7294
Time elapsed: 5.76 seconds
Epoch 56 | Loss: 2.7209
Time elapsed: 5.86 seconds
Epoch 57 | Loss: 2.7121
Time elapsed: 5.96 seconds
Epoch 58 | Loss: 2.7029
Time elapsed: 6.06 seconds
Epoch 59 | Loss: 2.6932
Time elapsed: 6.16 seconds
Epoch 60 | Loss: 2.6830
Time elapsed: 6.26 seconds
Epoch 61 | Loss: 2.6724
Time elapsed: 6.36 seconds
Epoch 62 | Loss: 2.6613
Time elapsed: 6.46 seconds
Epoch 63 | Loss: 2.6500
Time elapsed: 6.56 seconds
Epoch 64 | Loss: 2.6383
Time elapsed: 6.67 seconds
Epoch 65 | Loss: 2.6265
Time elapsed: 6.77 seconds
Epoch 66 | Loss: 2.6145
Time elapsed: 6.87 seconds
Epoch 67 | Loss: 2.6025
Time elapsed: 6.97 seconds
Epoch 68 | Loss: 2.5904
Time elapsed: 7.07 seconds
Epoch 69 | Loss: 2.5780
```

```
Time elapsed: 7.17 seconds
Epoch 70 | Loss: 2.5655
Time elapsed: 7.27 seconds
Epoch 71 | Loss: 2.5525
Time elapsed: 7.37 seconds
Epoch 72 | Loss: 2.5390
Time elapsed: 7.47 seconds
Epoch 73 | Loss: 2.5248
Time elapsed: 7.57 seconds
Epoch 74 | Loss: 2.5099
Time elapsed: 7.67 seconds
Epoch 75 | Loss: 2.4941
Time elapsed: 7.77 seconds
Epoch 76 | Loss: 2.4775
Time elapsed: 7.87 seconds
Epoch 77 | Loss: 2.4601
Time elapsed: 7.98 seconds
Epoch 78 | Loss: 2.4421
Time elapsed: 8.08 seconds
Epoch 79 | Loss: 2.4235
Time elapsed: 8.18 seconds
Epoch 80 | Loss: 2.4044
Time elapsed: 8.28 seconds
Epoch 81 | Loss: 2.3850
Time elapsed: 8.38 seconds
Epoch 82 | Loss: 2.3652
Time elapsed: 8.48 seconds
Epoch 83 | Loss: 2.3449
Time elapsed: 8.59 seconds
Epoch 84 | Loss: 2.3239
Time elapsed: 8.69 seconds
Epoch 85 | Loss: 2.3021
Time elapsed: 8.79 seconds
Epoch 86 | Loss: 2.2795
Time elapsed: 8.89 seconds
Epoch 87 | Loss: 2.2561
Time elapsed: 8.99 seconds
Epoch 88 | Loss: 2.2318
Time elapsed: 9.09 seconds
Epoch 89 | Loss: 2.2069
Time elapsed: 9.19 seconds
Epoch 90 | Loss: 2.1815
Time elapsed: 9.29 seconds
Epoch 91 | Loss: 2.1555
Time elapsed: 9.39 seconds
Epoch 92 | Loss: 2.1291
Time elapsed: 9.49 seconds
Epoch 93 | Loss: 2.1025
```

```
Time elapsed: 9.59 seconds
Epoch 94 | Loss: 2.0757
Time elapsed: 9.69 seconds
Epoch 95 | Loss: 2.0489
Time elapsed: 9.79 seconds
Epoch 96 | Loss: 2.0224
Time elapsed: 9.89 seconds
Epoch 97 | Loss: 1.9965
Time elapsed: 9.99 seconds
Epoch 98 | Loss: 1.9714
Time elapsed: 10.09 seconds
Epoch 99 | Loss: 1.9468
Time elapsed: 10.19 seconds
Epoch 100 | Loss: 1.9230
Time elapsed: 10.29 seconds
Epoch 101 | Loss: 1.9001
Time elapsed: 10.39 seconds
Epoch 102 | Loss: 1.8783
Time elapsed: 10.49 seconds
Epoch 103 | Loss: 1.8574
Time elapsed: 10.59 seconds
Epoch 104 | Loss: 1.8377
Time elapsed: 10.69 seconds
Epoch 105 | Loss: 1.8193
Time elapsed: 10.79 seconds
Epoch 106 | Loss: 1.8021
Time elapsed: 10.90 seconds
Epoch 107 | Loss: 1.7861
Time elapsed: 11.00 seconds
Epoch 108 | Loss: 1.7713
Time elapsed: 11.10 seconds
Epoch 109 | Loss: 1.7576
Time elapsed: 11.20 seconds
Epoch 110 | Loss: 1.7450
Time elapsed: 11.30 seconds
Epoch 111 | Loss: 1.7333
Time elapsed: 11.40 seconds
Epoch 112 | Loss: 1.7223
Time elapsed: 11.51 seconds
Epoch 113 | Loss: 1.7118
Time elapsed: 11.61 seconds
Epoch 114 | Loss: 1.7017
Time elapsed: 11.71 seconds
Epoch 115 | Loss: 1.6919
Time elapsed: 11.82 seconds
Epoch 116 | Loss: 1.6823
Time elapsed: 11.92 seconds
Epoch 117 | Loss: 1.6728
```

```
Time elapsed: 12.02 seconds
Epoch 118 | Loss: 1.6633
Time elapsed: 12.12 seconds
Epoch 119 | Loss: 1.6539
Time elapsed: 12.22 seconds
Epoch 120 | Loss: 1.6447
Time elapsed: 12.32 seconds
Epoch 121 | Loss: 1.6356
Time elapsed: 12.42 seconds
Epoch 122 | Loss: 1.6269
Time elapsed: 12.52 seconds
Epoch 123 | Loss: 1.6183
Time elapsed: 12.62 seconds
Epoch 124 | Loss: 1.6102
Time elapsed: 12.72 seconds
Epoch 125 | Loss: 1.6025
Time elapsed: 12.82 seconds
Epoch 126 | Loss: 1.5952
Time elapsed: 12.92 seconds
Epoch 127 | Loss: 1.5883
Time elapsed: 13.02 seconds
Epoch 128 | Loss: 1.5818
Time elapsed: 13.12 seconds
Epoch 129 | Loss: 1.5757
Time elapsed: 13.23 seconds
Epoch 130 | Loss: 1.5699
Time elapsed: 13.33 seconds
Epoch 131 | Loss: 1.5645
Time elapsed: 13.43 seconds
Epoch 132 | Loss: 1.5593
Time elapsed: 13.53 seconds
Epoch 133 | Loss: 1.5543
Time elapsed: 13.63 seconds
Epoch 134 | Loss: 1.5496
Time elapsed: 13.73 seconds
Epoch 135 | Loss: 1.5450
Time elapsed: 13.83 seconds
Epoch 136 | Loss: 1.5405
Time elapsed: 13.93 seconds
Epoch 137 | Loss: 1.5361
Time elapsed: 14.03 seconds
Epoch 138 | Loss: 1.5319
Time elapsed: 14.13 seconds
Epoch 139 | Loss: 1.5277
Time elapsed: 14.23 seconds
Epoch 140 | Loss: 1.5237
Time elapsed: 14.33 seconds
Epoch 141 | Loss: 1.5197
```

```
Time elapsed: 14.43 seconds
Epoch 142 | Loss: 1.5158
Time elapsed: 14.53 seconds
Epoch 143 | Loss: 1.5123
Time elapsed: 14.63 seconds
Epoch 144 | Loss: 1.5088
Time elapsed: 14.73 seconds
Epoch 145 | Loss: 1.5054
Time elapsed: 14.83 seconds
Epoch 146 | Loss: 1.5022
Time elapsed: 14.93 seconds
Epoch 147 | Loss: 1.4991
Time elapsed: 15.03 seconds
Epoch 148 | Loss: 1.4961
Time elapsed: 15.13 seconds
Epoch 149 | Loss: 1.4931
Time elapsed: 15.23 seconds
Epoch 150 | Loss: 1.4903
Time elapsed: 15.33 seconds
Epoch 151 | Loss: 1.4875
Time elapsed: 15.44 seconds
Epoch 152 | Loss: 1.4848
Time elapsed: 15.54 seconds
Epoch 153 | Loss: 1.4820
Time elapsed: 15.64 seconds
Epoch 154 | Loss: 1.4793
Time elapsed: 15.74 seconds
Epoch 155 | Loss: 1.4767
Time elapsed: 15.84 seconds
Epoch 156 | Loss: 1.4742
Time elapsed: 15.95 seconds
Epoch 157 | Loss: 1.4717
Time elapsed: 16.05 seconds
Epoch 158 | Loss: 1.4693
Time elapsed: 16.15 seconds
Epoch 159 | Loss: 1.4669
Time elapsed: 16.25 seconds
Epoch 160 | Loss: 1.4645
Time elapsed: 16.35 seconds
Epoch 161 | Loss: 1.4621
Time elapsed: 16.45 seconds
Epoch 162 | Loss: 1.4597
Time elapsed: 16.56 seconds
Epoch 163 | Loss: 1.4574
Time elapsed: 16.66 seconds
Epoch 164 | Loss: 1.4550
Time elapsed: 16.76 seconds
Epoch 165 | Loss: 1.4527
```

```
Time elapsed: 16.86 seconds
Epoch 166 | Loss: 1.4504
Time elapsed: 16.96 seconds
Epoch 167 | Loss: 1.4482
Time elapsed: 17.06 seconds
Epoch 168 | Loss: 1.4460
Time elapsed: 17.16 seconds
Epoch 169 | Loss: 1.4438
Time elapsed: 17.26 seconds
Epoch 170 | Loss: 1.4417
Time elapsed: 17.36 seconds
Epoch 171 | Loss: 1.4397
Time elapsed: 17.46 seconds
Epoch 172 | Loss: 1.4376
Time elapsed: 17.57 seconds
Epoch 173 | Loss: 1.4356
Time elapsed: 17.67 seconds
Epoch 174 | Loss: 1.4337
Time elapsed: 17.77 seconds
Epoch 175 | Loss: 1.4318
Time elapsed: 17.87 seconds
Epoch 176 | Loss: 1.4299
Time elapsed: 17.97 seconds
Epoch 177 | Loss: 1.4281
Time elapsed: 18.07 seconds
Epoch 178 | Loss: 1.4263
Time elapsed: 18.17 seconds
Epoch 179 | Loss: 1.4245
Time elapsed: 18.27 seconds
Epoch 180 | Loss: 1.4228
Time elapsed: 18.37 seconds
Epoch 181 | Loss: 1.4212
Time elapsed: 18.48 seconds
Epoch 182 | Loss: 1.4196
Time elapsed: 18.58 seconds
Epoch 183 | Loss: 1.4180
Time elapsed: 18.68 seconds
Epoch 184 | Loss: 1.4165
Time elapsed: 18.78 seconds
Epoch 185 | Loss: 1.4150
Time elapsed: 18.88 seconds
Epoch 186 | Loss: 1.4135
Time elapsed: 18.98 seconds
Epoch 187 | Loss: 1.4121
Time elapsed: 19.08 seconds
Epoch 188 | Loss: 1.4108
Time elapsed: 19.18 seconds
Epoch 189 | Loss: 1.4094
```

```
Time elapsed: 19.29 seconds
Epoch 190 | Loss: 1.4081
Time elapsed: 19.39 seconds
Epoch 191 | Loss: 1.4069
Time elapsed: 19.49 seconds
Epoch 192 | Loss: 1.4056
Time elapsed: 19.59 seconds
Epoch 193 | Loss: 1.4044
Time elapsed: 19.69 seconds
Epoch 194 | Loss: 1.4033
Time elapsed: 19.79 seconds
Epoch 195 | Loss: 1.4021
Time elapsed: 19.89 seconds
Epoch 196 | Loss: 1.4010
Time elapsed: 19.99 seconds
Epoch 197 | Loss: 1.4000
Time elapsed: 20.09 seconds
Epoch 198 | Loss: 1.3990
Time elapsed: 20.19 seconds
Epoch 199 | Loss: 1.3980
Time elapsed: 20.29 seconds
Epoch 200 | Loss: 1.3970
Time elapsed: 20.39 seconds
Epoch 201 | Loss: 1.3961
Time elapsed: 20.49 seconds
Epoch 202 | Loss: 1.3952
Time elapsed: 20.60 seconds
Epoch 203 | Loss: 1.3942
Time elapsed: 20.70 seconds
Epoch 204 | Loss: 1.3934
Time elapsed: 20.80 seconds
Epoch 205 | Loss: 1.3925
Time elapsed: 20.90 seconds
Epoch 206 | Loss: 1.3916
Time elapsed: 21.00 seconds
Epoch 207 | Loss: 1.3907
Time elapsed: 21.10 seconds
Epoch 208 | Loss: 1.3898
Time elapsed: 21.20 seconds
Epoch 209 | Loss: 1.3890
Time elapsed: 21.30 seconds
Epoch 210 | Loss: 1.3881
Time elapsed: 21.40 seconds
Epoch 211 | Loss: 1.3873
Time elapsed: 21.50 seconds
Epoch 212 | Loss: 1.3865
Time elapsed: 21.60 seconds
Epoch 213 | Loss: 1.3857
```

```
Time elapsed: 21.70 seconds
Epoch 214 | Loss: 1.3850
Time elapsed: 21.80 seconds
Epoch 215 | Loss: 1.3842
Time elapsed: 21.91 seconds
Epoch 216 | Loss: 1.3835
Time elapsed: 22.01 seconds
Epoch 217 | Loss: 1.3828
Time elapsed: 22.12 seconds
Epoch 218 | Loss: 1.3821
Time elapsed: 22.22 seconds
Epoch 219 | Loss: 1.3814
Time elapsed: 22.32 seconds
Epoch 220 | Loss: 1.3808
Time elapsed: 22.42 seconds
Epoch 221 | Loss: 1.3801
Time elapsed: 22.52 seconds
Epoch 222 | Loss: 1.3795
Time elapsed: 22.62 seconds
Epoch 223 | Loss: 1.3789
Time elapsed: 22.72 seconds
Epoch 224 | Loss: 1.3783
Time elapsed: 22.82 seconds
Epoch 225 | Loss: 1.3777
Time elapsed: 22.92 seconds
Epoch 226 | Loss: 1.3771
Time elapsed: 23.02 seconds
Epoch 227 | Loss: 1.3766
Time elapsed: 23.12 seconds
Epoch 228 | Loss: 1.3760
Time elapsed: 23.22 seconds
Epoch 229 | Loss: 1.3755
Time elapsed: 23.32 seconds
Epoch 230 | Loss: 1.3750
Time elapsed: 23.42 seconds
Epoch 231 | Loss: 1.3745
Time elapsed: 23.52 seconds
Epoch 232 | Loss: 1.3740
Time elapsed: 23.62 seconds
Epoch 233 | Loss: 1.3735
Time elapsed: 23.73 seconds
Epoch 234 | Loss: 1.3730
Time elapsed: 23.83 seconds
Epoch 235 | Loss: 1.3726
Time elapsed: 23.93 seconds
Epoch 236 | Loss: 1.3721
Time elapsed: 24.03 seconds
Epoch 237 | Loss: 1.3717
```

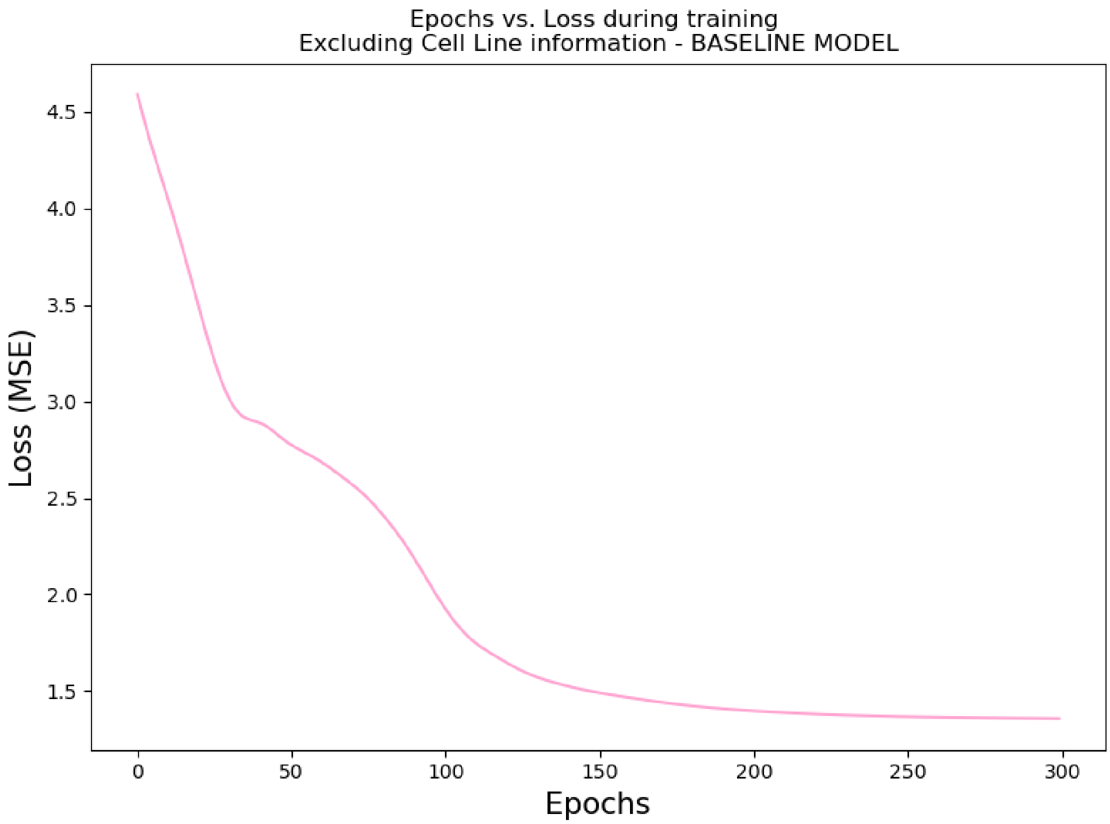
```
Time elapsed: 24.13 seconds
Epoch 238 | Loss: 1.3713
Time elapsed: 24.23 seconds
Epoch 239 | Loss: 1.3708
Time elapsed: 24.33 seconds
Epoch 240 | Loss: 1.3704
Time elapsed: 24.43 seconds
Epoch 241 | Loss: 1.3700
Time elapsed: 24.53 seconds
Epoch 242 | Loss: 1.3697
Time elapsed: 24.63 seconds
Epoch 243 | Loss: 1.3693
Time elapsed: 24.73 seconds
Epoch 244 | Loss: 1.3689
Time elapsed: 24.83 seconds
Epoch 245 | Loss: 1.3686
Time elapsed: 24.93 seconds
Epoch 246 | Loss: 1.3682
Time elapsed: 25.03 seconds
Epoch 247 | Loss: 1.3679
Time elapsed: 25.13 seconds
Epoch 248 | Loss: 1.3675
Time elapsed: 25.23 seconds
Epoch 249 | Loss: 1.3672
Time elapsed: 25.33 seconds
Epoch 250 | Loss: 1.3669
Time elapsed: 25.44 seconds
Epoch 251 | Loss: 1.3666
Time elapsed: 25.54 seconds
Epoch 252 | Loss: 1.3663
Time elapsed: 25.64 seconds
Epoch 253 | Loss: 1.3660
Time elapsed: 25.74 seconds
Epoch 254 | Loss: 1.3657
Time elapsed: 25.84 seconds
Epoch 255 | Loss: 1.3654
Time elapsed: 25.94 seconds
Epoch 256 | Loss: 1.3652
Time elapsed: 26.04 seconds
Epoch 257 | Loss: 1.3649
Time elapsed: 26.15 seconds
Epoch 258 | Loss: 1.3646
Time elapsed: 26.25 seconds
Epoch 259 | Loss: 1.3644
Time elapsed: 26.35 seconds
Epoch 260 | Loss: 1.3641
Time elapsed: 26.45 seconds
Epoch 261 | Loss: 1.3639
```

```
Time elapsed: 26.55 seconds
Epoch 262 | Loss: 1.3637
Time elapsed: 26.65 seconds
Epoch 263 | Loss: 1.3634
Time elapsed: 26.75 seconds
Epoch 264 | Loss: 1.3632
Time elapsed: 26.85 seconds
Epoch 265 | Loss: 1.3630
Time elapsed: 26.95 seconds
Epoch 266 | Loss: 1.3628
Time elapsed: 27.05 seconds
Epoch 267 | Loss: 1.3625
Time elapsed: 27.16 seconds
Epoch 268 | Loss: 1.3623
Time elapsed: 27.25 seconds
Epoch 269 | Loss: 1.3621
Time elapsed: 27.35 seconds
Epoch 270 | Loss: 1.3619
Time elapsed: 27.45 seconds
Epoch 271 | Loss: 1.3617
Time elapsed: 27.55 seconds
Epoch 272 | Loss: 1.3615
Time elapsed: 27.65 seconds
Epoch 273 | Loss: 1.3613
Time elapsed: 27.75 seconds
Epoch 274 | Loss: 1.3611
Time elapsed: 27.86 seconds
Epoch 275 | Loss: 1.3609
Time elapsed: 27.96 seconds
Epoch 276 | Loss: 1.3607
Time elapsed: 28.05 seconds
Epoch 277 | Loss: 1.3606
Time elapsed: 28.15 seconds
Epoch 278 | Loss: 1.3604
Time elapsed: 28.25 seconds
Epoch 279 | Loss: 1.3602
Time elapsed: 28.36 seconds
Epoch 280 | Loss: 1.3601
Time elapsed: 28.46 seconds
Epoch 281 | Loss: 1.3599
Time elapsed: 28.56 seconds
Epoch 282 | Loss: 1.3598
Time elapsed: 28.66 seconds
Epoch 283 | Loss: 1.3596
Time elapsed: 28.76 seconds
Epoch 284 | Loss: 1.3595
Time elapsed: 28.86 seconds
Epoch 285 | Loss: 1.3593
```

```
Time elapsed: 28.96 seconds
Epoch 286 | Loss: 1.3592
Time elapsed: 29.06 seconds
Epoch 287 | Loss: 1.3590
Time elapsed: 29.16 seconds
Epoch 288 | Loss: 1.3589
Time elapsed: 29.26 seconds
Epoch 289 | Loss: 1.3588
Time elapsed: 29.36 seconds
Epoch 290 | Loss: 1.3586
Time elapsed: 29.47 seconds
Epoch 291 | Loss: 1.3585
Time elapsed: 29.57 seconds
Epoch 292 | Loss: 1.3584
Time elapsed: 29.68 seconds
Epoch 293 | Loss: 1.3582
Time elapsed: 29.78 seconds
Epoch 294 | Loss: 1.3581
Time elapsed: 29.88 seconds
Epoch 295 | Loss: 1.3580
Time elapsed: 29.98 seconds
Epoch 296 | Loss: 1.3579
Time elapsed: 30.09 seconds
Epoch 297 | Loss: 1.3577
Time elapsed: 30.19 seconds
Epoch 298 | Loss: 1.3576
Time elapsed: 30.29 seconds
Epoch 299 | Loss: 1.3575
Time elapsed: 30.39 seconds
```

[69]:

```
epochs_vs_loss_baseline(my_epochs, my_losses)
```

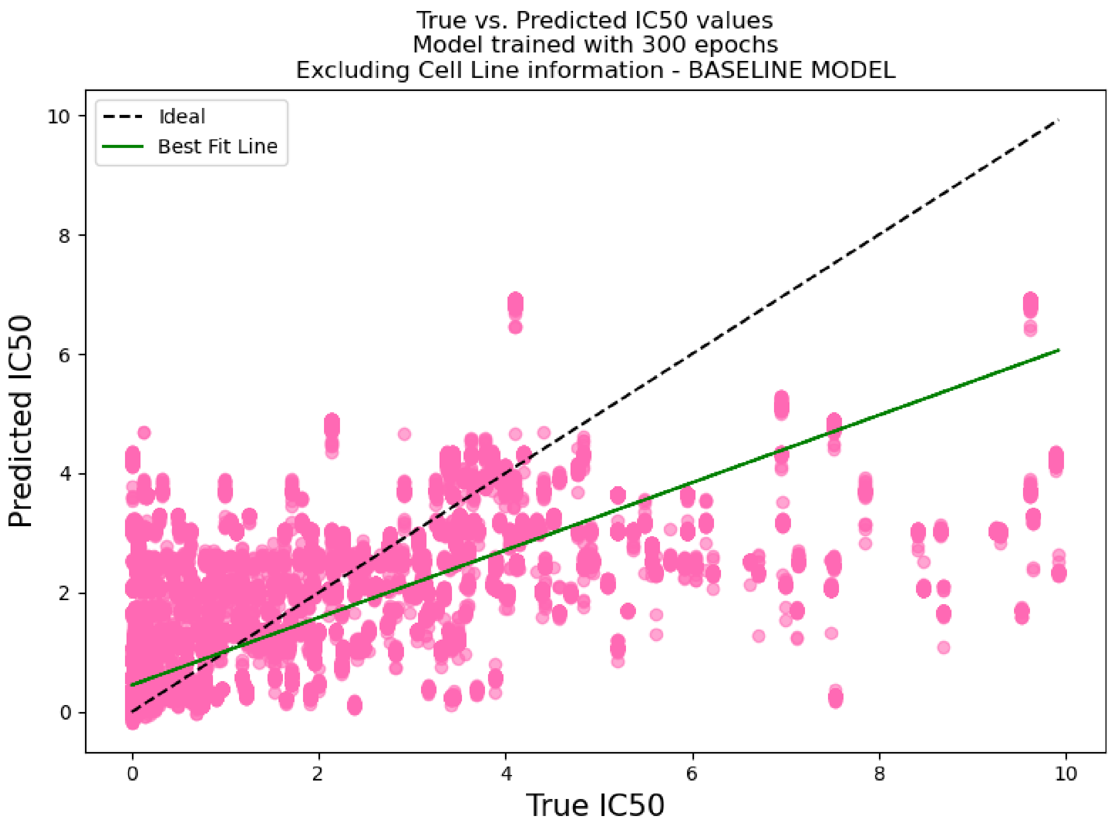


[70]:
```
drug_features_test = X_test[['Molecular Mass', 'LogP', 'NumHDonors',␣
↪'NumHAcceptors', 'TPSA']]
patient_features_test = X_test.drop(columns=drug_features_test.columns)

metrics = evaluate_model(model, drug_features_test, patient_features_test,␣
↪y_test)

print(f"MSE: {metrics[0]:.4f}")
print(f"MAE: {metrics[1]:.4f}")
print(f"R²: {metrics[2]:.4f}")
```

```
MSE: 1.3475
MAE: 0.5955
R²: 0.5657
```

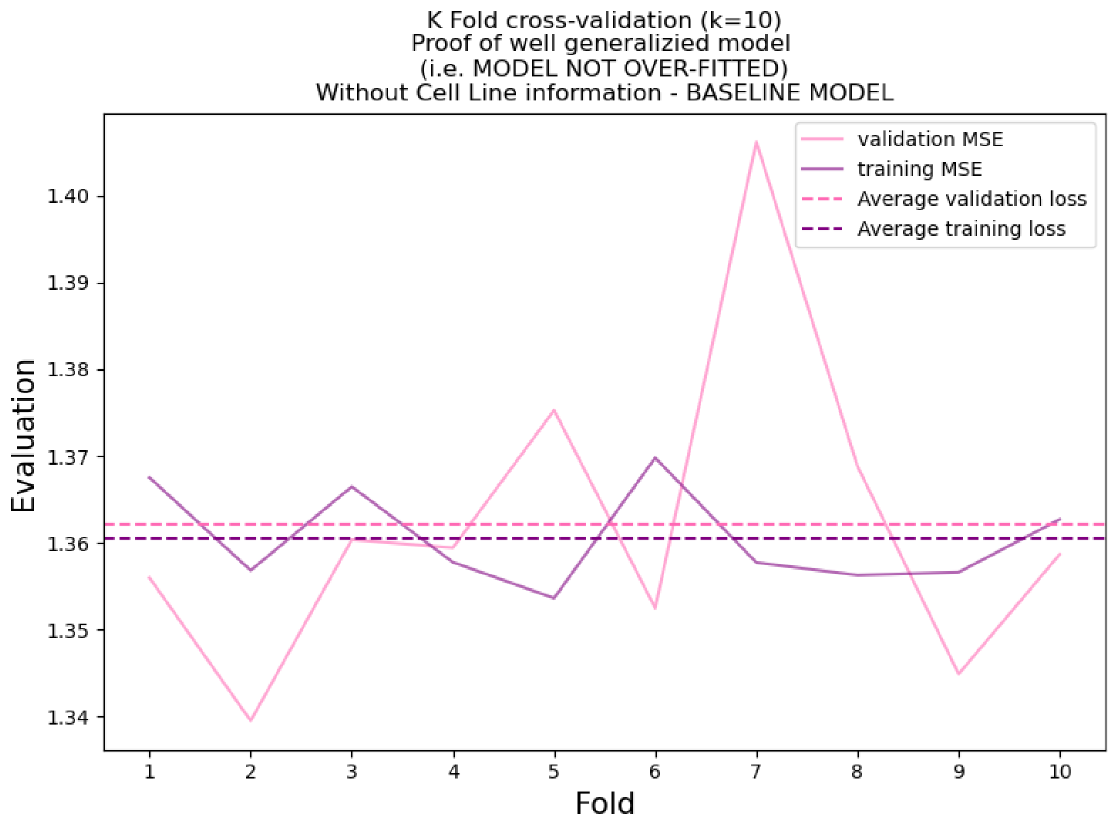
[71]:
```
with torch.no_grad():
    preds = model(torch.tensor(drug_features_test.values).float(),
                  torch.tensor(patient_features_test.values).float()).cpu().
↪numpy()
plot_pred_vs_true_baseline(y_test.to_numpy(), preds)
```

```
/Users/joyceyu/Documents/CHEM277B - ML Algorithms/Final
project/Data_277_-Cancer/CHEM277B_functions.py:176: UserWarning: color is
redundantly defined by the 'color' keyword argument and the fmt string "r--" (->
color='r'). The keyword argument will take precedence.
  plt.plot([true.min(), true.max()], [true.min(), true.max()], 'r--',
label='Ideal', color = "Black")
```



[72]:
```
k_fold_cv = compute_CV_error(X_train,y_train)
```

[74]:
```
Kfold_CV_plot_baseline(k_fold_cv)
```


K Fold cross-validation (k=10)
Proof of well generalizied model
(i.e. MODEL NOT OVER-FITTED)
Without Cell Line information - BASELINE MODEL
validation MSE
training MSE
Average validation loss
Average training loss
1.40
1.39
1.38
1.37
1.36
1.35
1.34
Evaluation
1
2
3
4
5
6
7
8
9
10
Fold

```
[ ]:
```