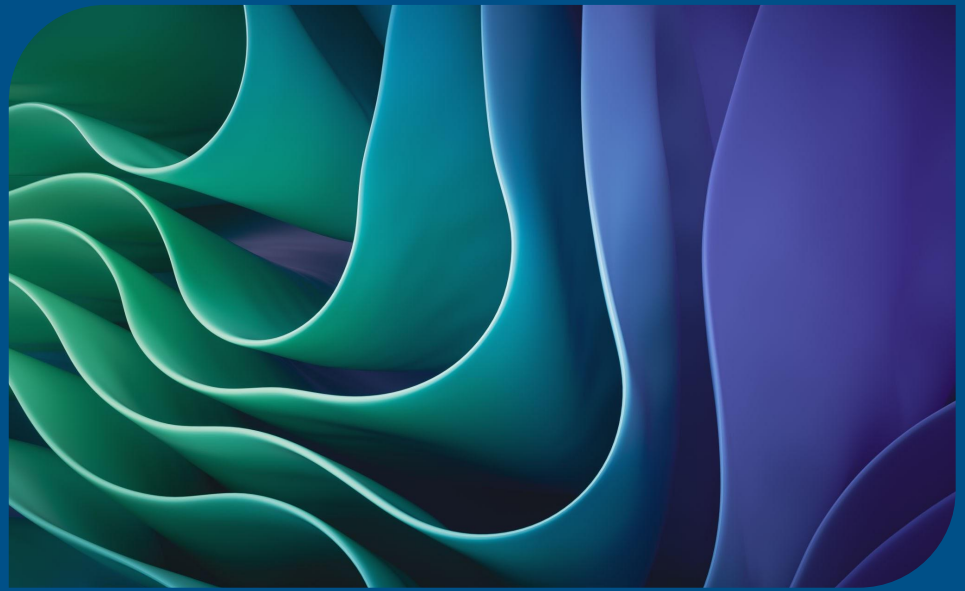
MM.DD.YY

Predicting Kinase Selectivity Using Machine Learning

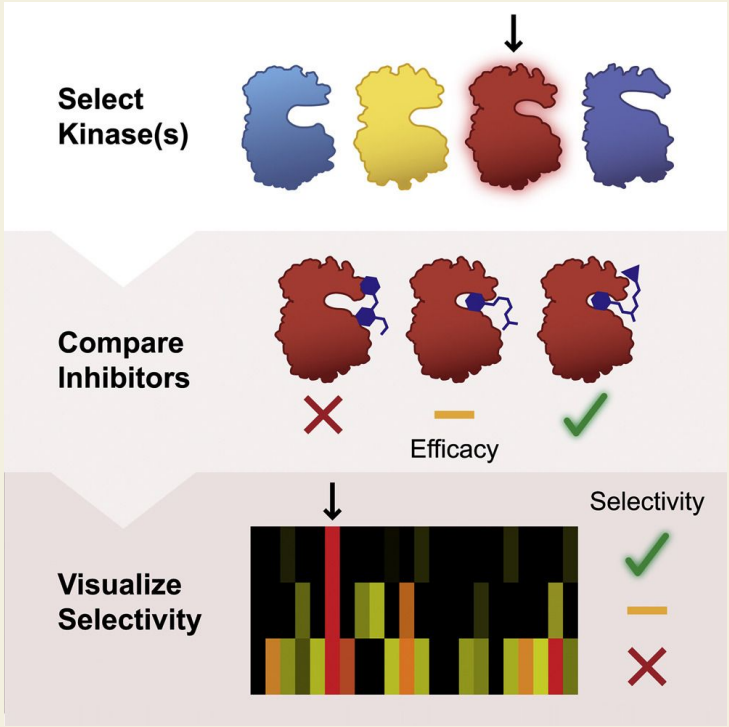
Challenge 1: Molecular Forecaster



Lorem Ipsum

Challenge

- Kinases
- Inhibitors
- Importance of Selectivity
  - Off-target effects
    - Side effects & drug effectiveness
  - Diversity of Kinase families



https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.cell.com%2Fiscience%2Ffulltext%2FS2589-0042%252818%252930144-5&psig=AOvVaw0xSWY7vw22IaPmlKYz83kr&ust=1742229396109000&source=images&cd=vfe&opi=89978449&ved=0CBEQjRxqFwoTCOjx-rWEj4wDFQAAAAAdAAAAABAE

Strategy

- Classification
- Model: XGBoost Classifier
- Features used for model training
- SMILES representations

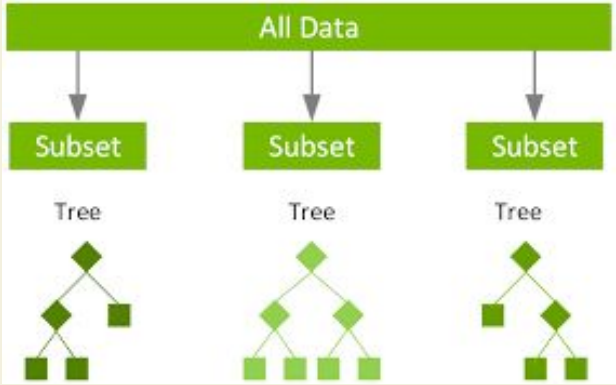
Accession Number	Entrez Gene Symbol	Kinase	Mutant	Kinase Group
NP_055726.3	AAK1	AAK1	NO	Other
NP_005148.2	ABL1	ABL1(E255K)-phosphorylated	YES	TK
NP_005148.2	ABL1	ABL1(F317I)-nonphosphorylated	YES	TK
NP_005148.2	ABL1	ABL1(F317I)-phosphorylated	YES	TK
NP_005148.2	ABL1	ABL1(F317L)-nonphosphorylated	YES	TK

Accession Number	Entrez Gene Symbol	Kinase	A-674563	AB-1010	ABT-869	AC220	AG-013736	AST-487	AT-7519	AZD-1152HQPA	AZD-2171	AZD-6244/ARRY-886	BI-2536	BIBW-2992
NP_055726.	AAK1	AAK1	43	10001	10001	10001	1200	10001	10001	3000	10001	10001	2800	10001
NP_005148.	ABL1	ABL1(E255K)-phosphorylated	10001	140	10001	10001	63	75	10001	9600	65	10001	5900	420

	Compound	SMILES	Binding Mode (based on ABL1-phos. vs. -nonphos affinity)	S(300nM)	S(3000nM)
2	A-674563	CC1=C2C=C(C=CC2=NN1)C3=CC(=CN=C3)OCC(CC4=CC=CC=C4)N	undetermined	0.1166	0.2772
3	AB-1010	CC1=C(C=C(C=C1)NC(=O)C2=CC=C(C=C2)CN3CCN(CC3)C)NC4=NC(=CS4)C5=CN=CC=C5	Type II	0.0337	0.0622
4	ABT-869	CC1=CC(=C(C=C1)F)NC(=O)NC2=CC=C(C=C2)C3=C4C(=CC=C3)NN=C4N	undetermined	0.0648	0.1839
5	AC220	CC(C)(C)C1=CC(=NO1)NC(=O)NC2=CC=C(C=C2)C3=CN4C5=C(C=C(C=C5)OCCN6CCOCC6)SC4=N3	Type II	0.0285	0.0751
6	AG-013736	CNC(=O)C1=CC=CC=C1SC2=CC3=C(C=C2)C(=NN3)C=CC4=CC=CC=N4	Type I	0.057	0.1969
7	AST-487	CCN1CCN(CC1)CC2=C(C=C(C=C2)NC(=O)NC3=CC=C(C=C3)OC4=NC=NC(=C4)NC)C(F)(F)F	Type II	0.2617	0.4922
8	AT-7519	C1CNCCC1NC(=O)C2=C(C=NN2)NC(=O)C3=C(C=CC=C3Cl)Cl	undetermined	0.0674	0.0933
9	AZD-1152HQPA	CCN(CCCOC1=CC2=C(C=C1)C(=NC=N2)NC3=NNC(=C3)CC(=O)NC4=CC(=CC=C4)F)CCO	Type II	0.0311	0.114

Justifications

- Why XGBoost instead of other models?
- Our evaluation metrics: ML accuracy



5.2.2 CLASSIFICATION APPROACH

- **MSE** loss to evaluate S accuracy : $MSE = \frac{1}{n}\sum_{i=1}^{n}(S_i - \hat{S}_i)^2$
- **Zero-One Loss** to evaluate classification accuracy (for both 300nM and 3000nM thresholds) : $L(K_{d\text{pred}}, K_d) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\left(K_{d\text{pred}}^{(i)} < 3000 \neq K_d{}^{(i)} < 3000\right)$

```python
# returns a dataframe with each inhibitor OR kinase and the corresponding generat
def makeFeature(sourcedf, fxns:dict, type:{"inhibitor", "kinase"}):
    ''' fxns is a dictionary where:
        keys: column name, of the target column in sourcedf
        values: function, to apply to the target column in sourcedf and generate
                each feature value is then normalized
    '''
    featuredf = pd.DataFrame()
    for col,fxn in fxns.items():
        if type == "inhibitor":
            featuredf['Compound'] = sourcedf['Compound']
        else:
            featuredf['Kinase'] = sourcedf['Kinase']
        featuredf[col+" embedded"] = sourcedf[col].apply(fxn)
        featuredf[col+" embedded"] = MinMaxScaler().fit_transform(np.array(featur
    return featuredf
```

```python
# returns an dataframe with both the inhibitor and kinase features for each poss
def createX(inhibitordf, kinasedf):
    result = pd.merge(inhibitordf, kinasedf, how='cross')
    return result

# returns a dataframe with the Kd for each possible inhibitor-kinase pair
def createY(sourcedf, Sthresh):
    Kdf = sourcedf.drop(['Accession Number', 'Entrez Gene Symbol'], axis=1)
    Kdf = Kdf.set_index('Kinase')
    stacked = Kdf.stack().reset_index()
    stacked.columns = ['Kinase', 'Compound', 'Kd']

    stacked[f'Kd'] = (stacked['Kd'] < Sthresh).astype(int)
    return stacked.set_index(['Compound', 'Kinase'])
```

```python
def XGBtrain(X, Y, title):
    data = X.merge(Y, on=['Compound', 'Kinase'], how='inner')
    x = data.iloc[:,:-1].values
    y = data.iloc[:,-1].values

    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size

    # X_train, X_val, y_train, y_val = train_test_split(X_train, y_train
    # the above is not needed because there are 12 inhibitors that will
    # for the challenge at 11:00 am on 3/16/2025

    model = XGBClassifier()
    model.fit(X_train, y_train)
    y_train = np.array(y_train).ravel()
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred) * 100
```

```python
def selectivity(df, col, threshold):
    filtered_col = df[col][df[col] < threshold]

    sum = filtered_col.sum()
    count = filtered_col.count()

    return sum/count

# example on the original Kd dataset given
selectivity(Kd, 'AB-1010', 3000)
```

Attempts

Implementation of SMILES string derived info using RDKit Fingerprint

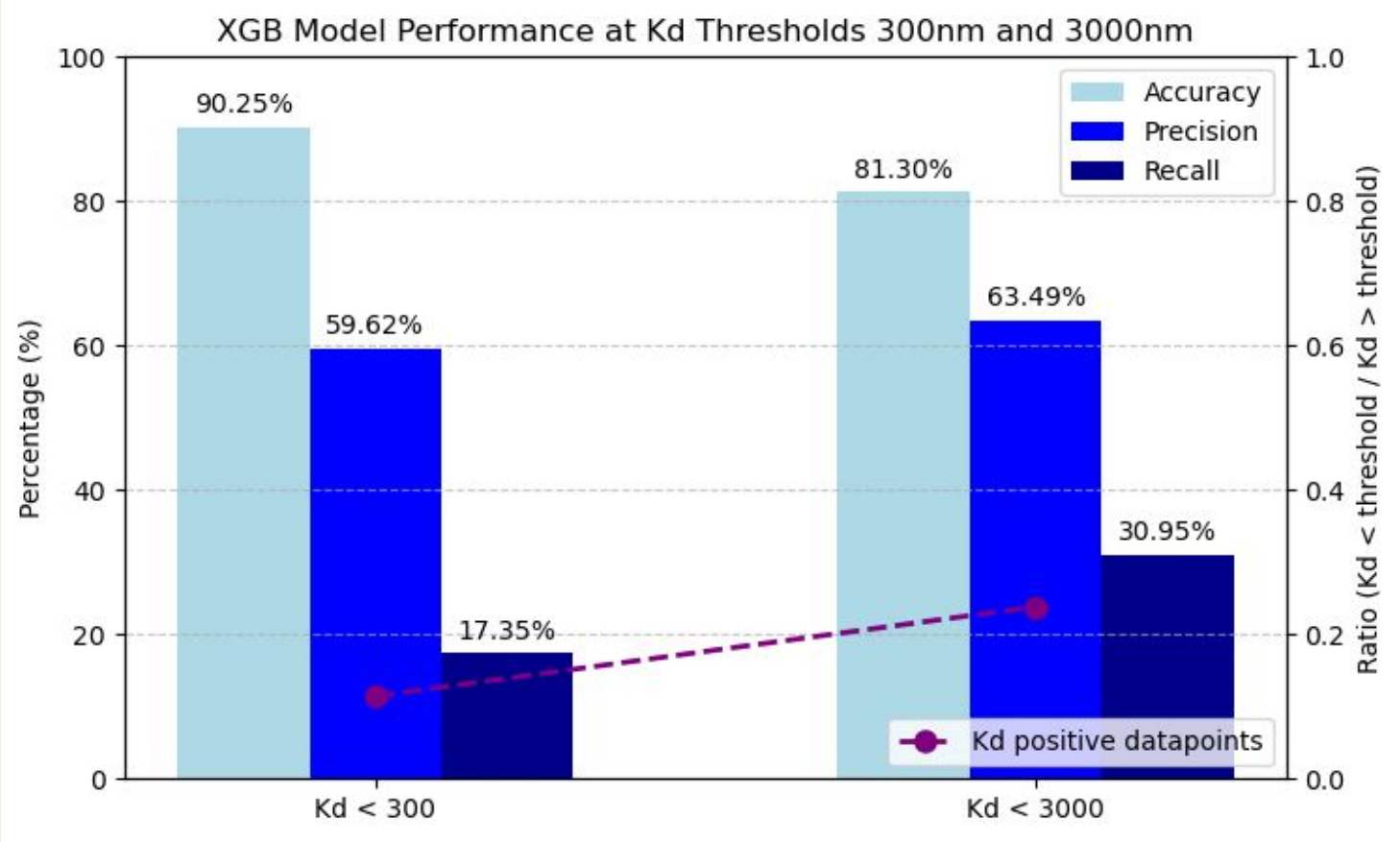
```
inhibitorFeatures = makeFeature(sensitivities_mod, {"S(300nM)":(lambda a: a),
                                                    "S(3000nM)":(lambda a: a),
                                                    "HD":(lambda a: a),
                                                    "HA":(lambda a: a),
                                                    "AR":(lambda a: a),
                                                    "RB":(lambda a: a),
                                                    "TPSA":(lambda a: a),
                                                    "LogP":(lambda a: a),
                                                    "MW":(lambda a: a)}, "inhibitor")
```

✓ 0.0s

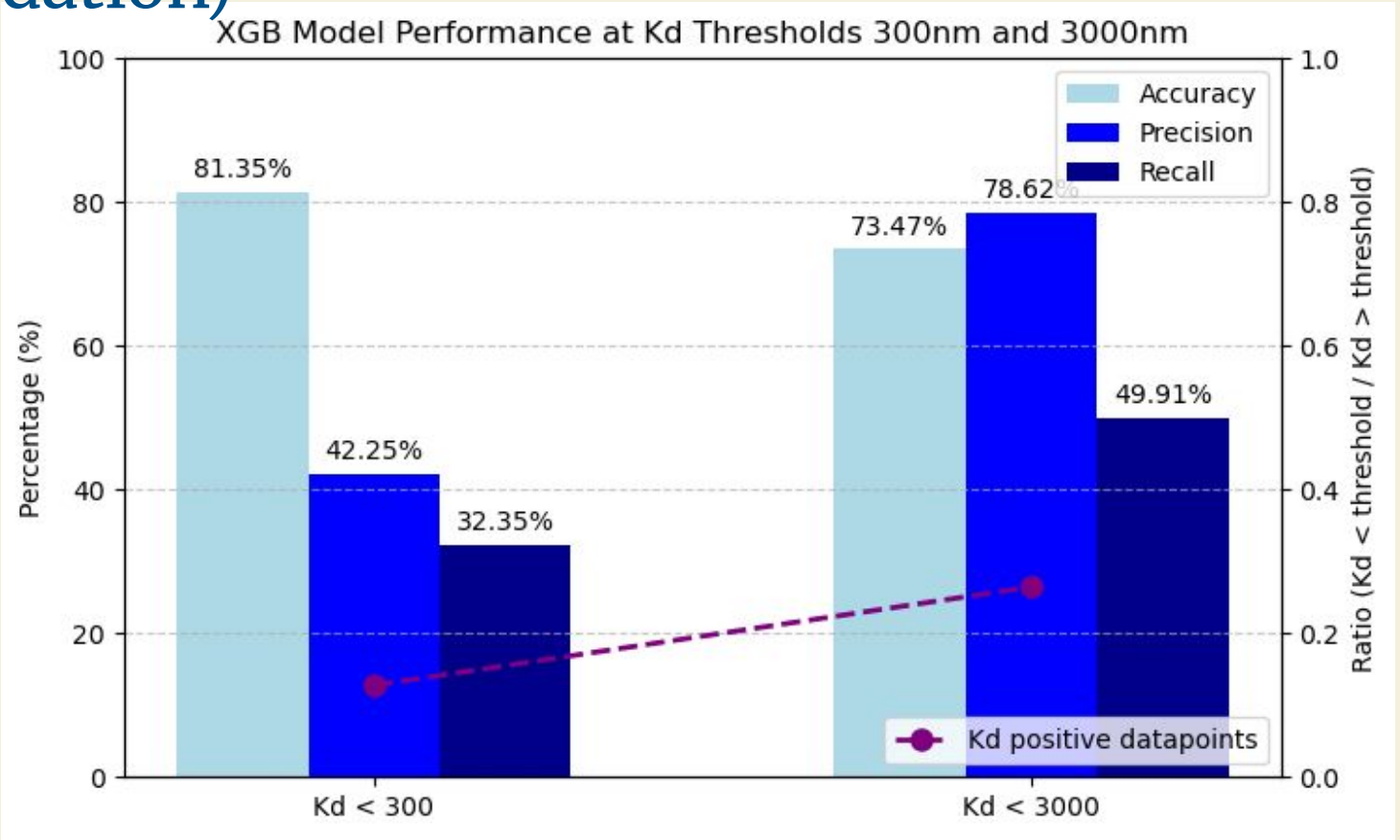
```
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit import DataStructs
from rdkit.Chem import Descriptors

def get_fingerprint(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:  # Check for invalid molecules
        return None

    fpgen = AllChem.GetMorganGenerator(radius=2, fpSize=2048)
    fps = fpgen.GetFingerprint(mol)

    HD = Descriptors.NumHDonors(mol)
    HA = Descriptors.NumHAcceptors(mol)
    AR = Descriptors.NumAromaticRings(mol)
    RB = Descriptors.NumRotatableBonds(mol)
    TPSA = Descriptors.TPSA(mol)
    LogP = Descriptors.MolLogP(mol)
    MW = Descriptors.MolWt(mol)

    return {
        #'Fingerprint': fps,
        'HD': HD,
        'HA': HA,
        'AR': AR,
        'RB': RB,
        'TPSA': TPSA,
        'LogP': LogP,
        'MW': MW
    }

fingerprint_columns = sensitivities["SMILES"].apply(get_fingerprint).apply(pd.Series)
sensitivities_mod = pd.concat([sensitivities, fingerprint_columns], axis=1)
```

✓ 0.0s

Results: 20% of the original data (test set)



Results: predicting on the hidden test set (validation)



Future Considerations

- Alternative models like Random Forest or deep learning models like Graph Neural Networks (GNNs)
- Other molecular descriptors (like physicochemical properties)
- Visualizing the data using Cytoscape
- Measuring performance metrics using K-fold cross validation