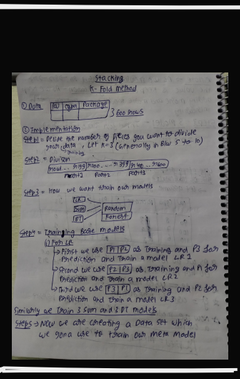
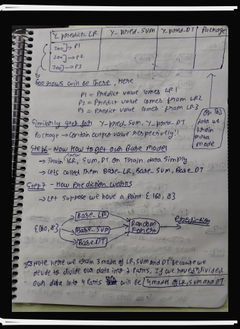
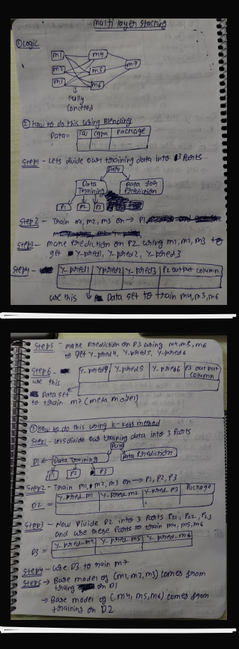
Stacking and Blending Ensembles

- video
- github
- core intuition
  - first we train multiple models(called Base models) on data, then we give these base model output to a new model(called meta model) as input with data output column
  - FOR EXAMPLE
    - data -> iq | cgpa | package
    - train 3 models -> SVM , DT , LR
      - Prediction columns from these 3 models
      - **Y_pred_1 , Y_pred_2 , Y_pred_3**
    - then
      - Train a new model -> Random Forest
        - input columns for this model = Y_pred_1 , Y_pred_2 , Y_pred_3
        - output column for this model = Package
  - For a new Point -> how prediction works
    - First base model create **Y_pred_1 , Y_pred_2 , Y_pred_3**
    - Bases of these values meta model do prediction
- Implementation





- Implementation
- How to avoid overfitting
  - core intuition
    - There is very high probability that the complete model start overfitting
  - 2 methods to avoid over fitting
    - Blending/Hold out method
      - core intuition
        - Divide the data into 2 parts (Data training and Data test)
        - Divide the training data into 2 parts (D1 and D2)
        - use D1 to train base models
        - Then base models do prediction for D2 Y_pred1 , Y_pred2 , Y_pred3
        - These Y_pred1 , Y_pred2 , Y_pred3 and output column used for training meta model
      - not supported by sklearn , do it manually
    - Stacking/K-fold method
      - core intuition
        - here k= number of pieces you want to divide your data (Generally people take value in between 5-10)
        - Subtopic 2
      - supported by sklearn
- Multi layer stacking



- code
  - classification
    - Note here cv= cross_validation = act as k for k-fold mean
    - Subtofrom sklearn.ensemble import RandomForestClassifier
      from sklearn.neighbors import KNeighborsClassifier
      from sklearn.linear_model import LogisticRegression
      from sklearn.ensemble import GradientBoostingClassifier

```
estimators = [
    ('rf', RandomForestClassifier(n_estimators=10, random_state=42)),
    ('knn', KNeighborsClassifier(n_neighbors=10)),
    ('gbdt',GradientBoostingClassifier())
]
```

```
from sklearn.ensemble import
StackingClassifier

clf = StackingClassifier(
    estimators=estimators,
    final_estimator=LogisticRegression(),
    cv=10
)
```

```
clf.fit(X_train, y_train)
```

  - Regression

```
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import StackingRegressor
```

```
base_models = [
    ('lr', LinearRegression()),
    ('rf', RandomForestRegressor(n_estimators=10, random_state=42)),
    ('gb', GradientBoostingRegressor(n_estimators=10, random_state=42))
]
```

```
stacked_model = StackingRegressor(
    estimators=base_models,
    final_estimator=LinearRegression()
)
```

```
stacked_model.fit(X_train, y_train)
```