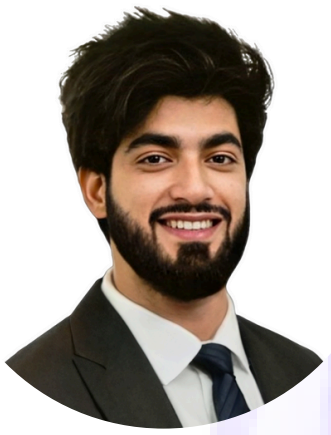

Tajamul Khan

Machine Learning Cheat Sheet

@Tajamulkhann

Types of Machine Learning

SUPERVISED LEARNING

Regression
- Linear
- Polynomial
- Ridge ,
- Lasso ,
- Decision Tree,
- Random Forest,
- SVR
- XGBoost

Classification
- Logistic
- Decision Tree
- Random Forest
- SVM
- k-NN
- Naïve Bayes
- XGBoost
- Neural Networks

UNSUPERVISED LEARNING

Clustering
- k-Means,
- Hierarchical
- DBSCAN,
- GMM
- Mean Shift

Dimensionality
- k-Means,
- Hierarchical
- DBSCAN,
- GMM
- Mean Shift

REINFORCEMENT LEARNING

Reward-based learning
- Q-Learning
- Deep Q-Networks (DQN)
- SARSA (State-Action-Reward-State-Action)
- Policy Gradient Methods
- Actor-Critic Methods
- Proximal Policy Optimization (PPO)
- Trust Region Policy Optimization (TRPO)
- Monte Carlo Methods

@Tajamulkhann
@Tajamul.datascientist

Data Preprocessing

df.isnull().sum(): Check missing values
df.dropna(): Remove missing values
df.fillna(value): Fill missing values
StandardScaler(): Standardization (mean=0, std=1)
MinMaxScaler(): Normalization (range [0,1])
LabelEncoder(): Convert categorical labels to numbers
OneHotEncoder(): Convert categorical features into dummy variables

@Tajamulkhann
@Tajamul.datascientist

Train Test Split

```
from sklearn.model_selection
import train_test_split


X_train, X_test, y_train,
y_test = train_test_split(X,
y, test_size=0.2,
random_state=42)
```


@Tajamulkhann
@Tajamul.datascientist

Regression Models

- Linear Regression
- Polynomial Regression
- Ridge & Lasso Regression
- Decision Tree Regression
- Random Forest Regression
- Support Vector Regression (SVR)

```
from sklearn.linear_model import
LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```


@Tajamulkhann
@Tajamul.datascientist

Classification Models

- Logistic Regression
- K-Nearest Neighbors (KNN)
- Decision Trees
- Random Forest
- Support Vector Machines (SVM)
- Naive Bayes
- Neural Networks (MLP)

```
from sklearn.ensemble import
RandomForestClassifier
model =
RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```


@Tajamulkhann
@Tajamul.datascientist

Evaluation Metrics

Regression

- **MAE (Mean Absolute Error):** Measures avg absolute error
- **MSE (Mean Squared Error):** Penalizes large errors
- **RMSE (Root Mean Squared Error):** Square root of MSE
- **R^2 Score:** Explains variance captured by model

Classification

- **Accuracy:** (TP + TN) / (TP + FP + TN + FN)
- **Precision:** TP / (TP + FP)
- **Recall:** TP / (TP + FN)
- **F1-Score:** 2 * (Precision * Recall) / (Precision + Recall)
- **ROC Curve:** Trade-off between TPR & FPR

@Tajamulkhann
@Tajamul.datascientist

Feature Selection & Engineering

Feature Importance: Use models like RandomForest
PCA (Principal Component Analysis): Reduce dimensionality
Correlation Matrix: Find highly correlated features


```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
```

@Tajamulkhann
@Tajamul.datascientist

Hyperparameter Tuning

GridSearchCV – Tries all combinations
RandomizedSearchCV – Randomized search

```
from sklearn.model_selection import
GridSearchCV
params = {'n_estimators': [50, 100,
150]}
grid =
GridSearchCV(RandomForestClassifier()
, params, cv=5)
grid.fit(X_train, y_train)
```


@Tajamulkhann
@Tajamul.datascientist

Clustering Algorithms

- K-Means Clustering
- Hierarchical Clustering
- DBSCAN (Density-Based Spatial Clustering)

```
from sklearn.cluster import
KMeans
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
labels = kmeans.predict(X)
```


@Tajamulkhann
@Tajamul.datascientist

Deep Learning Basics

- Neural Networks
- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks (RNNs)
- Transformers (BERT, GPT)

```
import tensorflow as tf
model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam',
loss='binary_crossentropy', metrics=['accuracy'])
```


@Tajamulkhann
@Tajamul.datascientist

Deep Learning Basics

- Neural Networks
- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks (RNNs)
- Transformers (BERT, GPT)

```
import tensorflow as tf
model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam',
loss='binary_crossentropy', metrics=['accuracy'])
```

@Tajamulkhann
@Tajamul.datascientist

Found Helpful ?

Repost

@Tajamulkhann

@Tajamul.datascientist

Follow for more!